AD-A053 477 GENERAL ELECTRIC CO DAYTONA BEACH FLA F/G 5/9
COMPUTER IMAGE GENERATION IMAGERY IMPROVEMENT: CIRCLES, CONTOUR--ETC(U)
SEP 77 W M BUNKER, N E FERRIS F33615-76-C-0038
UNCLASSIFIED AFHRL-TR-77-66 NL

1 OF 2
AD A053477

AFHRL-TR-77-66

(2)

AIR FORCE HUMAN RESOURCES LABORATORY

COMPUTER IMAGE GENERATION IMAGERY IMPROVEMENT:

CIRCLES, CONTOURS, AND TEXTURE

By

W. Marvin Bunker
Norman E. Ferris
General Electric Company
P.O. Box 2500
Daytona Beach, Florida 32015

ADVANCED SYSTEMS DIVISION
Wright-Patterson Air Force Base, Ohio 45433

September 1977
Final Report for Period June 1976 – August 1977

AIR FORCE SYSTEMS COMMAND
BROOKS AIR FORCE BASE, TEXAS 78235

NOTICE

When U.S. Government drawings, specifications, or other data are used for any purpose other than a definitely related Government procurement operation, the Government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise, as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This final report was submitted by General Electric Company, Daytona Beach, Florida 32015, under contract F33615-76-C-0038, project 6114, with Advanced Systems Division, Air Force Human Resources Laboratory (AFSC), Wright-Patterson Air Force Base, Ohio 45433. Lt Michael L. Ingalls, Simulation Techniques Branch, was the contract monitor.

This report has been reviewed and cleared for open publication and/or public release by the appropriate Office of Information (OI) in accordance with AFR 190-17 and DoDD 5230.9. There is no objection to unlimited distribution of this report to the public at large, or by DDC to the National Technical Information Service (NTIS).

This technical report has been reviewed and is approved for publication.

GORDON A. ECKSTRAND, Director
Advanced Systems Division

DAN D. FULGHAM, Colonel, USAF
Commander

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFHRL-TR-77-66	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): COMPUTER IMAGE GENERATION IMAGERY IMPROVEMENT: CIRCLES, CONTOURS, AND TEXTURE	5. TYPE OF REPORT & PERIOD COVERED: Final rept. Jun 1976 – Aug 1977
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): W. Marvin Bunker, Norman E. Ferris	8. CONTRACT OR GRANT NUMBER(s): F33615-76-C-0038
9. PERFORMING ORGANIZATION NAME AND ADDRESS: General Electric Company, P.O. Box 2500, Daytona Beach, Florida 32015	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 62205F, 6114052 3
11. CONTROLLING OFFICE NAME AND ADDRESS: HQ Air Force Human Resources Laboratory (AFSC), Brooks Air Force Base, Texas 78235	12. REPORT DATE: Sept 1977
	13. NUMBER OF PAGES: 136 (12) 137 p.
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Advanced Systems Division, Air Force Human Resources Laboratory, Wright-Patterson Air Force Base, Ohio 45433	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

(16) 611 (17) 05

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

computer image generation (CIG)
contour simulation
scene texture
velocity cues
visual simulation systems

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Computer Image Generation (CIG) visual simulation systems are now being procured for research and as pilot training simulators. Current systems generate scenes that do not provide sufficient velocity and altitude cues. Training effectiveness would be increased by improving this aspect of the visual scenes.

This report covers investigations into three approaches to improving visual scene cues. These are generation of contours (ridge-like and valley-like features), generation of circular features, and a ground-map technique for generating surface texture. In each of the three areas, the approach was to develop efficient algorithms, to produce scenes for evaluation of the algorithms and illustration of various methods of using the capability, and to estimate

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

148 300 JOB

Item 20 Continued:

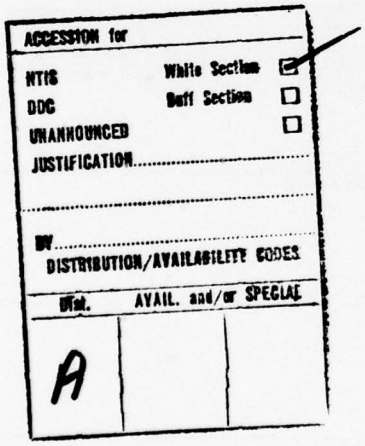
requirements for hardware implementation in real-time systems. A goal in all cases was to minimize the use of the edge-processing capacity of the CIG system. In several cases, scenes were prepared to evaluate the effect of applying different algorithms to achieve a desired result.

Two quite different algorithms were developed to produce circular features on the surface. One proved superior both in results and in reduced processing requirements. Very minor modification to the algorithm would allow simulation of nonsurface circles, ellipses and of spheres. This capability was added to the software scene generation facility. These features can be used to enhance scene realism by simulating lakes, settling tanks, curved roads, puffy clouds, etc. Scenes were produced to illustrate this capability. Further extension will allow ellipsoid simulation.

For reference and comparison with other techniques, contours were generated using standard CIG edges to form three-dimensional objects. It was determined that, within certain constraints, identical effects could be produced using delineators defined entirely on the surface. To achieve spatial consistency, it is necessary to redefine these delineators for each scene. This added computation is not justified by reduction in requirements elsewhere in the processing; further, the constraints would prevent contours produced in this manner from providing valid cues near the ground. Effort was then applied to evaluating a number of techniques for producing contours using edges.

The effort on the ground-map texture generation was not performed under this contract. It was an IR&D effort, and the results are included in this report by agreement with the Air Force. It involves defining a set of maps on the surface, with algorithms to determine the map contribution to each pixel during scene generation. Evaluation scenes indicate this may be very effective in providing texture cues.

In addition to the scenes included in this report, video tapes were made using the static scene generator with a time-lapse video disc recorder. These show the dynamic effect of the techniques developed.



SUMMARY

PROBLEM

In Computer Image Generation (CIG) systems, the fundamental unit of capacity is the edge. This is the term applied to the straight line segments bounding the faces that form objects and surface features in a scene. The limited number of edges available in a given system are generally concentrated in several areas of primary interest, such as targets, airports, or carriers. When flying in regions away from these concentrations of detail, a pilot finds the texture and features in the scene insufficient to provide motion cues, attitude cues, and altitude cues.

A related problem area affects scene realism. In an effort to conserve edges, scene features are frequently modeled as a nonrealistic approximation of the actual feature. Consider a surface circle, part of a practice bombing target. The actual circle can be approximated as closely as desired by using a sufficient number of edges but, for edge saving, a less close approximation is used. Or assume a ridge is modeled to give sensitive altitude cues when flying near the ground. A ridge formed of a large number of edges can be quite realistic. A minimum-edge ridge will give valid cues, but lack realism.

To solve the problem stated above, it is necessary to develop techniques for simulation of essential cues realistically either without use of edges or using edges with greater efficiency.

APPROACH

This contract called for effort to be applied to the study of two approaches to alleviating this problem. These were circular-feature simulation and contour simulation.

Circular Features

If circular features lying on the ground could be generated efficiently; with individually specified location, size, and color, they could be used in large numbers to provide surface detail in areas that have few other features. They could be used to assure that all portions of a flight would contain detail for motion cues.

The possibility exists that they could be effective in providing altitude cues. The most significant cues for visually estimating altitude are derived from

objects on the ground of known size. In a recent study of air-to-surface weapons delivery, [1]Figure 3, "Low Level Velocity and Altitude Cues" shows a scene in which objects (such as buses, cars, aircraft sitting on an apron) and buildings are modeled to serve this function. This can be effective, but it is expensive in terms of edges. Circular features, with size distribution approximating that of trees in an area, may serve this function for trainees.

The approach to evaluating the potential of the circular-feature capability was to develop algorithms for generating these features, verify their validity, select best algorithms after evaluation, and then produce a variety of data bases and evaluation scenes to demonstrate the methods in which they might be used and provide material for determining their effectiveness. Video tapes of moving sequences for use in evaluation of dynamic effects are also essential.

Contours

The starting point for the contour investigation was the known fact that tonal patterns generated on a flat surface can give the effect of relief, or a third dimension, on the surface. This has the potential of simulating contours more efficiently than detailed modeling of their shapes in three dimensions using edges. There are limitations associated with this approach—situations that cannot be validly simulated.

The approach was to compare the computational requirements of the three-dimensional edge approach with the flat-surface approach, verify the equivalency of the results when the flat-surface constraints were not violated, make a selection, and produce data bases and scenes using a variety of modeling techniques and parameter values. The goal is to provide material for evaluation of the relationship between simulation difficulty and simulation effectiveness of the selected approaches. As in the case of the circular features, video tapes are required for dynamic evaluation.

Ground-Map Texture

A closely related Independent Research and Development (IR&D) study effort proceeded concurrently with the early part of this contract. This involved defining a set of texture maps covering an infinite expanse of ground surface, determining for each pixel the contribution of the maps to its tone, and applying

[1]Eric G. Monroe, Robert W. Rife, Michael L. Cyrus, Lynn C. Thompson, AFHRL-TR-76-40, ASUPT Visual Simulation of Air-to-Surface Weapons Delivery, Flying Training Division, Williams Air Force Base, Arizona 85224, June 1976.

the result to surface background or surface faces, as specified by control bits associated with these faces. The approach, similar to the above, was to develop algorithms and apply them with various sets of parameters to produce evaluation scenes. The results of this study are being included in this report per prior agreement with the Air Force.

RESULTS

Elliptical and Spherical Features

The change in the label of this part of the effort is based on some of the results. As the algorithms were developed to meet the stated goal of producing circular features lying on the surface, it was found that minor modifications would (a) allow the defined features to be elliptical as well as circular, (b) allow the features to be defined at any elevation (not just on the surface) and with any orientation (not just horizontal), and (c) allow spherical features to be validly simulated.

Investigation of methods of using these features extended this part of the study well beyond the original concept of scattering them over the fields to provide detail. They can be used in a more deterministic manner to add realism to simulated visual scenes in the modeling of a variety of objects.

Investigation of hardware implementation demonstrated the feasibility of providing these features in real-time CIG systems.

Contours

The requirements for producing contours defined entirely on the surface were developed, and scenes were made demonstrating the equivalence of the results to those provided by contours defined in three dimensions. The constraints of the surface approach were defined, and it was determined they would prevent the use of this approach in low-altitude flight situations in which contour visual cues are most essential. Analysis of computational requirements indicated the load for the surface contours would be very close to that of fully defined contours—possibly even greater.

There are many ways in which contour cues can be provided using edges to form three-dimensional definitions. If a given feature is modeled with sufficient edges to give a rather faithful spatial representation and then the curvature algorithm incorporating the gradient algorithm is validly applied, extremely realistic scenes result. This was demonstrated in an earlier contract. This, however, requires a large number of edges.

A variety of more edge-efficient approaches were modeled and evaluated. This included minimum-edge definitions with and without the gradient algorithm, along with approaches using some additional edges to better meet the goal of contours with gentle curves and variable slope simulation.

A rather significant finding of these evaluations was the following. When the gradient algorithm is used in violation of the constraints of the full-curvature algorithm, in an attempt to simulate shapes significantly different from the spatially defined shapes of features, the probability of success is low.

However, minimum-edge contours without the gradient algorithm very effectively provide orientation and elevation cues, although they are lacking in realism.

Ground-Map Texture

Scenes made to illustrate ground-map texture showed very impressive results. Hardware estimates indicate that real-time implementation, while feasible, is definitely not inexpensive.

CONCLUSIONS

Elliptical and Spherical Features

These features can provide detail cues over large areas, and can add significantly to the realism with which a variety of features can be simulated. Real-time hardware implementation is quite feasible with the cost being a function of the number of features to be generated. A determination of the value to training of various numbers of these features is necessary as part of a decision on how many to specify in a system. The evaluation scenes produced during this study will help in making this determination.

Contours

The effort did not succeed in developing any technique for producing contour cues superior to the use of edges for this purpose. Models were made and scenes produced illustrating a number of methods of using edges with varying efficiency for contours. These scenes can provide preliminary guidance to help in producing such contours. These contours can be produced on any real-time system that includes the curvature algorithm, such as Advanced Simulator for Pilot Training (ASPT). Much more rapid and valid evaluation could be done on such a system.

Ground-Map Texture

The impressive results provided by the evaluation scenes produced strong indications that some implementation of ground-map texture would be a desirable addition to any full-capability CIG system. The rather high cost of the hardware estimated for implementation indicates additional effort in two areas.

Additional experimentation with methods of using the ground-map texture capability, along with production of evaluation scenes, will more clearly indicate its value to a training situation—will help more fully answer, "What are we buying for this money?"

More extensive work on the algorithms and on their hardware implementation may lower the anticipated cost below current estimates.

PREFACE

This work was performed under Contract Number F33615-76-C-0038.

This study was initiated by the Advanced Systems Division, Air Force Human Resources Laboratory, Wright-Patterson Air Force Base, Ohio. The effort was conducted by General Electric Company, Space Division, P. O. Box 2500, Daytona Beach, Florida 32015. Dr. W. Marvin Bunker was the principal investigator for the General Electric Company. Lt. Michael L. Ingalls, of the Simulation Techniques Branch, Advanced Systems Division, was the contract monitor for the Air Force Human Resources Laboratory.

CONTENTS

CONTENTS (Continued)

CONTENTS (Continued)

FIGURES

FIGURES (Continued)

TABLES

COMPUTER IMAGE GENERATION IMAGERY IMPROVEMENT: CIRCLES, CONTOURS, AND TEXTURE

SECTION 1

INTRODUCTION

The effort covered in this report is part of a continuing process directed toward improving results when using simulators in training applications requiring visual-scene cues. Computer image generation of these scenes has many characteristics very desirable for trainers. Unlimited gaming area, dynamic validity, precise scene generation allowing juxtaposed scenes for wide fields of view, ease of environment modification, and variety of special effects (such as fog and haze) are among these. However, computer image generation systems are still behind photographically based systems and camera-model systems in total scene detail to provide visual cues.

Improved cues can result from increasing the total number of edges that can be processed and displayed by a system, from using a given edge capability more efficiently, or fron non-edge based techniques for generation of texture detail or other scene features. Effort is underway in all three of the above. The study covered in this report was directed toward the third approach—algorithms to provide required scene detail by means other than directly modeling them with edges.

The activities and results are discussed in the following categories.

1.1 ELLIPTICAL AND SPHERICAL FEATURES

This includes discussion of two algorithms for generating video for simulation of the features. The basis for decision as to the preferable algorithm is given. Scenes illustrating varied application of the features are shown and discussed. Finally, the estimates for hardware for real-time implementation are covered.

1.2 CONTOURS

The details of the investigation that led to the abandonment of the non-edge approach to contours are covered. Data bases using edges with varying efficiency and using different approaches to tonal assignment are described. A number of scenes of these data bases are shown and discussed, along with the implications regarding preferred techniques for contour cue generation.

1.3 GROUND-MAP TEXTURE

The concept of ground-map texture is described, a number of the modes in which it might be used are discussed, and scenes illustrating the use of this technique are shown. Plans to extend the ground-map texture approach to non-surface faces are discussed.

1.4 FEATURE GENERATOR

This does not represent an assigned part of the study effort of the contract. It is a concept that came up during consideration of ways to most effectively use some of the capability developed during the study. It is also applicable to maximizing the efficiency of use of edges and is discussed in this context, as well as in connection with circular features and contours.

1.5 CONCLUSIONS AND RECOMMENDATIONS

The findings of this study and their significance are discussed. Recommendations for continuing effort leading toward the goal of maximum training effectiveness with CIG-produced visual cues, are included.

SECTION 2

ELLIPTICAL AND SPHERICAL FEATURES

2.1 APPROACH

The initial concept behind this portion of the study was the following. If small circular features on the surface can be generated efficiently without using system edge capacity, then they can be scattered in large numbers on portions of the ground surface currently lacking in texture cues, thus providing the missing cues.

The approach was to devise algorithms for the scene generation system to produce the circular features, to produce evaluation scenes and sequences to illustrate the effect of using the features in scenes, and to prepare hardware estimates for real-time implementation—with hardware as a function of quantity of features, if possible.

As effort proceeded, the major modification in the original plans was the determination that very little impact on algorithm complexity and hardware requirements would result if the types of features were expanded to include (a) ellipses, as well as circles; (b) circles and ellipses with any orientation and off-surface as well as on-surface; and (c) spheres, as well as flat features. Algorithms were modified to provide this extended capability. Hardware estimates were based on these full-capability algorithms. The programs were modified to include non-surface circles and spheres to allow scenes to be produced illustrating their use.

2.2 BRIEF DESCRIPTION OF ALGORITHMS

Under a projection transformation, circles, ellipses, and spheres not intersecting the view plane transform to view plane ellipses. Just as with edges, point lights, and other CIG scene detail, the task of the frame-rate processing (Frame 2) is to perform this transformation and to prepare the view plane description in such a manner as to minimize computational requirements in the faster line-rate and element-rate processing (Frame 3).

Two rather different Frame 3 algorithms were developed, programmed, and used for scene generation. They have somewhat different requirements for information content from Frame 2.

2.2.1 FRAME 2 PROCESSING

Points in the view window are defined by their values of I, scan line number, and J, element number along a scan line. The equation of any ellipse in the view window can be given as: $R1\,I^2 + R2\,J^2 + R3\,I\,J + R4\,I + R5\,J + R6 = 0$.

Frame 2 determines the coefficients, R1 through R6, for each ellipse and provides this information to Frame 3. It derives from these coefficients the extremum points of each ellipse and sends them to Frame 3: $I_{top}\ J_{top}$; $I_{bottom}\ J_{bottom}$; $I_{left}\ J_{left}$; $I_{right}\ J_{right}$. For one of the Frame 3 algorithms, Frame 2 also determines for each ellipse the vectors in the I-J plane defining the axes of the ellipse in this plane and the center of the ellipse. This is used in the Frame 3 distance algorithm, but not in the Frame 3 edge algorithm.

Frame 2 sends Frame 3 the above spatial definition of each ellipse along with its color and priority.

2.2.1.1 Ellipsoid Simulation

Frame 3 receives the definition of an ellipse, whether the original figure was an ellipse, a circle, a sphere, or any other feature which, under projection transformation, becomes an ellipse. In fact, the major portion of Frame 2 processing is independent of the type of source feature. Preliminary processing in Frame 2 transforms a sphere into an "equivalent ellipse definition." It is then processed as are all other ellipses.

Realism of modeling some features—trees, for example—would be improved by adding ellipsoids to the features that can be simulated. This can be easily done. It will involve adding provision to the data base for defining an ellipsoid and adding to the preliminary Frame 2 processing the steps to transform the ellipsoid to an equivalent ellipse. The remainder of the Frame 2 processing and the Frame 3 processing would remain unchanged.

2.2.1.2 Illumination Effects

There is no provision in the algorithm for the type of gradual tonal variation found in curvature simulation. Addition of illumination effects to entire features could be readily accomplished; just as it is now applied to flat faces to give orientation cues relative to illumination direction. It would be an early Frame 2 function, involving the dot product of the illumination vector with a face normal vector representing the elliptical feature, and tonal modification accordingly. For spheres and ellipsoids, the "face normal" would be defined for each new scene as part of the equivalent ellipse generation early in Frame 2.

2.2.2 FRAME 3 PROCESSING

For each scan line, Frame 3 must determine which ellipses contribute to that scan line, it must then organize the definitions of such "active" ellipses applicable to the current scan line and then determine the contribution of each ellipse to each pixel of the scan line. Two approaches were developed, programmed, and evaluated.

2.2.2.1 Frame 3 Distance Algorithm

The distance approach is based on the following concept. Figure 1 shows a portion of an ellipse passing through three pixels. For each such pixel, consider a ray from the center of the ellipse through the center of the pixel. Determine distance along this ray to the pixel center and to the boundary of the ellipse. Letting $\Delta D = D_{ellipse} - D_{pixel}$ (truncated to not exceed 0.5 in magnitude), and defining factor $F = 0.5 + \Delta D$ compute element color as F times ellipse color + (1 - F) times background color. Preliminary evaluation with curves of varying orientation and curvature indicated the results might be close enough to satisfactorily display the ellipses. For the figure, the sample results are shown in Table 1.

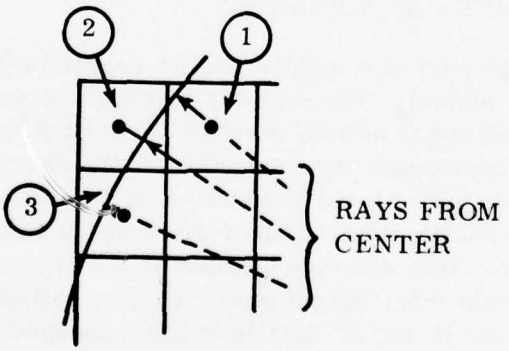


Figure 1. Distance Algorithm Concept

Table 1

Sample Results—Distance Approach

Pixel	ΔD	ΔD (truncated)	F	Color
1	0.6	0.5	1	1.0 Ellipse + 0 Background
2	-0.2	-0.2	0.3	0.3 Ellipse + 0.7 Background
3	0.15	0.15	0.65	0.65 Ellipse + 0.35 Background

The preliminary questions to be answered in connection with this approach were: (A) "Would it give a satisfactory representation of the desired features?", and (b) "Could algorithms be devised to obtain the required distances in a computationally efficient manner to make the approach feasible for implementation?" If answers to these two questions should be favorable, then other aspects of implementation (such as effective quantization smoothing for all combinations of features and edges) could be studied.

These questions will be discussed further in the section comparing the distance algorithm with the edge algorithm.

2.2.2.2 Frame 3 Edge Algorithm

Figure 2 shows part of a feature and its relation with two scan lines (pixels purposely not shown). The edge algorithm involves defining, for each scan line on which an ellipse is active, a set of temporary "edges" representing it on that scan line. These edges are formed from the extremum points of the ellipse and its intersection with the top and bottom of the current scan line. They are shown dashed in the illusttration. Once formed at the input to Frame 3 in a feature edge generator, they are then handled by the remainder of Frame 3 in a manner indistinguishable from the current-scan-line definition of actual scene edges. This has the merit that all details of this subsequent processing are mature and proven. It has the disadvantage that any per-scan-line edge processing capacity used for these features reduces that available for processing regular edges.

2.2.2.3 Comparison of Algorithms

2.2.2.3.1 Computational Load

The Major computational load in the distance algorithm is the determination of the distances from the ellipse centers to the pixel centers and the ellipse

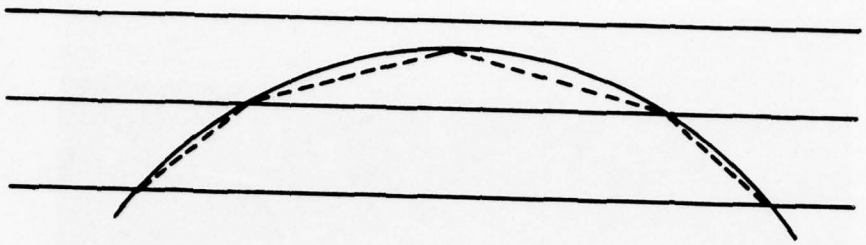


Figure 2. Edge Algorithm Concept

boundaries. It was hoped that some satisfactory approximation, relatively simple computationally, could be devised. Efforts to derive such an expression were not successful. Thus, the computational load for the distance algorithm is far greater than that for the edge approach.

2.2.2.3.2 Results

Both approaches form the required features.

An early question that arose in this study was how effectively circular features could be used to simulate various defined scene features, in addition to the originally intended scattered detail. Figure 3 shows a bomb circle as approximated on the ASPT system using edges. Figure 4 shows an approximation to this bomb circle produced using the edge algorithm. When the same scene was made using the distance algorithm, there was a barely perceptible difference consisting of some ragged regions around the edge. Analysis of the cause showed it to be inherent in the concept applied in the distance algorithm.

If the distance algorithm gave evidence of significant merit aside from the above problem, it is possible that some modifications could be devised to get around it. However, such did not seem to be the case, so subsequent effort and scene generation were all based on the circular-feature edge-generator concept.

2.2.2.4 Edge-Per-Scan-Line Loading

In many applications, addition of capability for elliptical- and spherical-feature simulation may require no increase in the edge-per-scan-line capacity, even when using the edge algorithm. A rather typical situation that has been noted on systems in the past is that in the vicinity of the airport, where large numbers of edges are used to define runways, markings, hangars, roads, etc., visual cues are adequate. Here, we might use a few of the new features for added realism. When flying away from the airport in regions where there are only a few large fields, the new features may be used in large numbers to provide visual

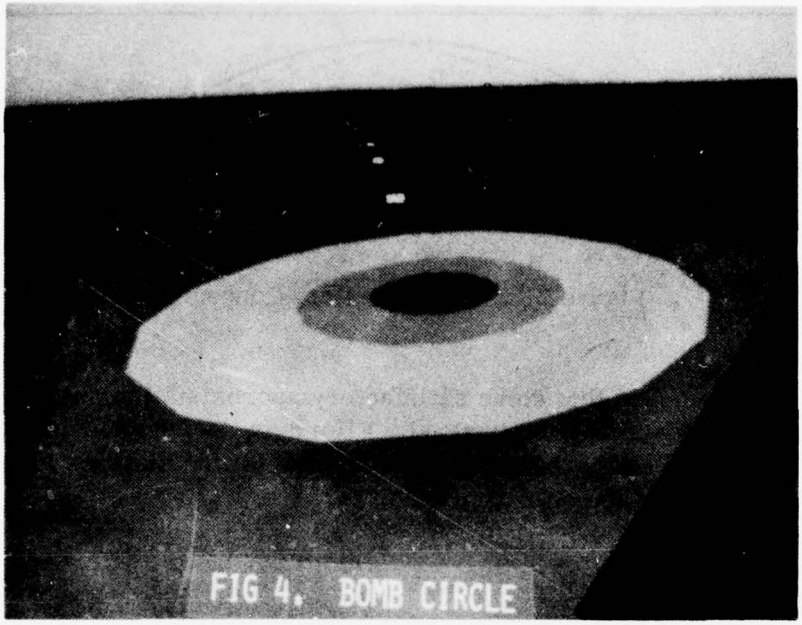


Figure 3. Bomb Circle Simulated With Edges

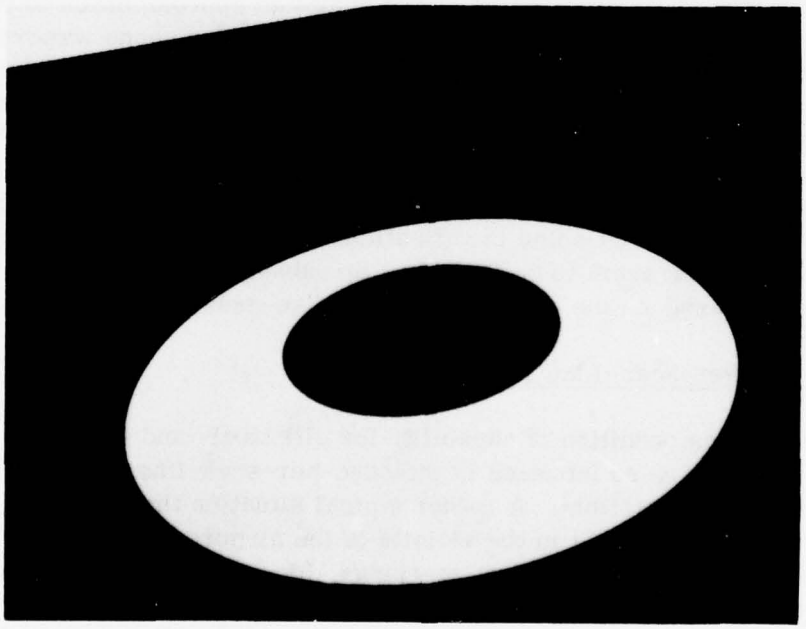


Figure 4. Bomb Circle Simulation Using Edge Algorithm

detail, and here very little of the edge-per-scan-line capacity is being used for scene edges.

2.3 EVALUATION SCENES

2.3.1 BOMB CIRCLE

Figure 4, discussed earlier, illustrates this application. The largest circle is the equivalent of a 472-edge approximation; yet, no portion of the processing is burdened with this many edges. Frame 2 gives Frame 3 instructions from which it generates edges for each scan line, as needed.

2.3.2 CYLINDRICAL TANKS

Figure 5, produced during an earlier contract, shows a group of oil storage tanks. Each tank is formed of 48 edges, 32 vertices, and 18 faces.

Figure 6 shows a similar group of tanks, each formed from two circular features and a single four-edge face.

A small amount of additional processing is required to obtain this efficiency in edge utilization. The face associated with the circular features must be redefined for each change in the viewpoint. The processing is simple. It would logically be done at the point where objects are extracted from environment memory and subjected to preliminary processing prior to the Frame 2 spatial transformations. At this point, channel assignment takes place, illumination angle effect on face colors is implemented, direction is tested for directional lights, etc. Also at this point, for cylinders, the "this frame" definition of the face would be computed. A test must also be made to determine if the viewpoint is above the cylinder or between the planes of the cylinder top and bottom. If the viewpoint is above, the top circle has "tank top" color or if the viewpoint is below the top, the top circle has "tank side" color.

2.3.3 CURVED ROAD AND CLOUD SEQUENCE

Figure 7 shows a sequence of views moving through a data base containing a cloud formed of a set of intersecting spheres. On the ground are two curved segments of a road that are produced by circular features with the same center and radii differing by the road width. Among other things, this sequence illustrates that the CIG priority system applies validly to these new features. The clouds are obscured by the prism when looking up, but they obscure the ground when looking down.

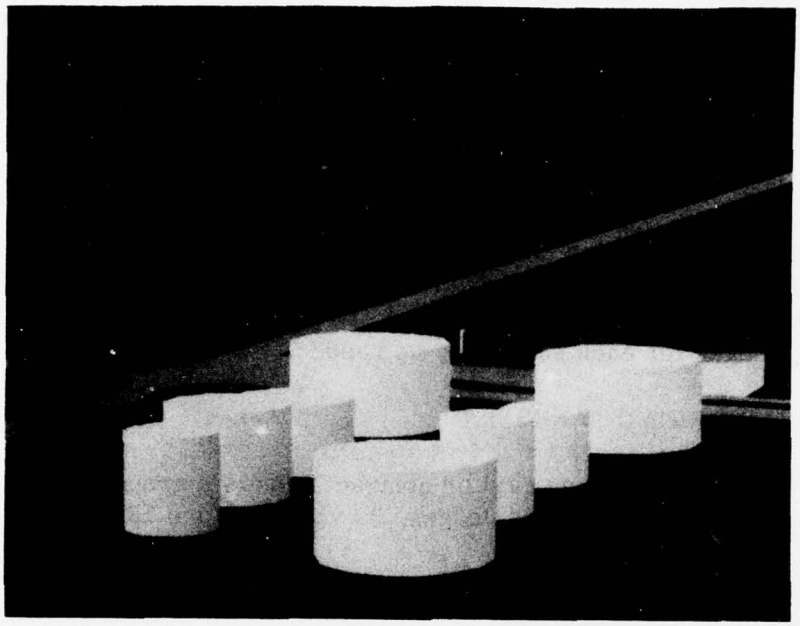


Figure 5. Cylindrical Tanks Simulated With Edges

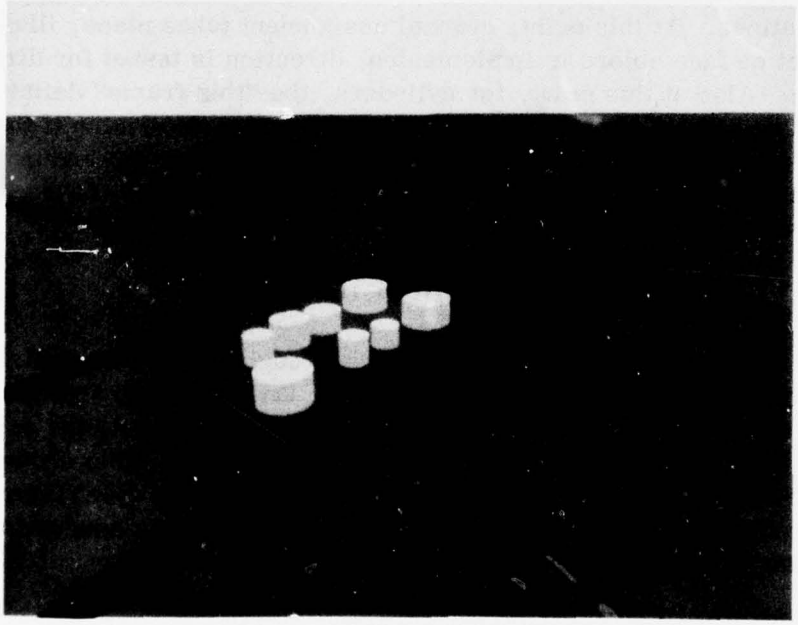
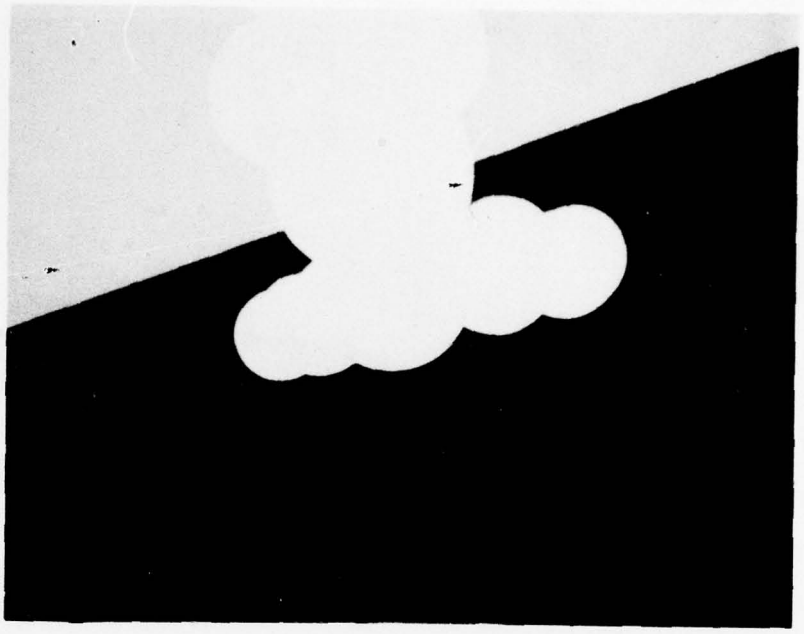


Figure 6. Cylindrical Tanks Simulated With Circular Features



(a)

(b)

Figure 7. Sequence Illustrating Clouds and Curved Road (Sheet 1 of 2)

(c)

(d)

Figure 7. Sequence Illustrating Clouds and Curved Road (Sheet 2 of 2)

2.3.4 RUNWAY SCENE SEQUENCE

Figure 8 shows scenes from a sequence through a data base designed to illustrate a number of possible applications for circular and spherical features. It includes juxtaposed circles simulating an irregularly shaped lake, isolated circles on the ground to simulate settling tanks, juxtaposed circles to represent tire marks on the runway, a sphere on a post representing a water tower, and groups of spherical features simulating trees. It may be that such trees will help provide the type of sensitive, low-altitude cues that the contour study is directed toward.

2.3.5 HIGH-DETAIL, CIRCULAR-FEATURE SEQUENCE

A data base was modeled containing regions with very few faces—the situation that has in the past made most apparent the need for additional detail. These regions were then enriched with large numbers of circular and spherical features, located in clumps of irregular size and shape. Since the primary function of this added detail is to provide the missing motion and change-of-altitude cues, critical evaluation must be done viewing the dynamic video tape made of the flight through this data base. The still scenes in Figure 9 are "snapshots" of points along the video tape sequence and give some indication of the results.

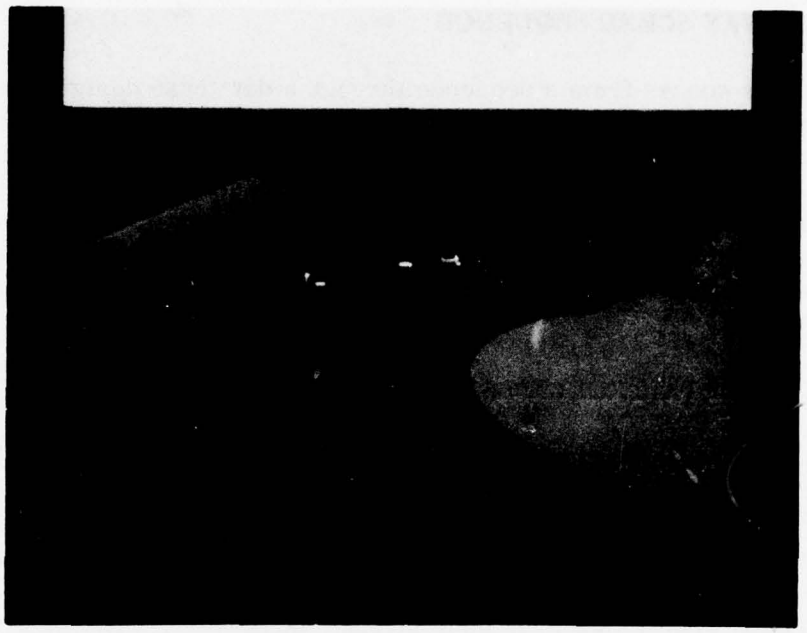
2.3.6 COMPOSITE SCENE

In practice, the new types of features developed in this study will be used in combination with edges to best achieve the desired combination of realism and effective visual cues for training. Figure 10, including a farmhouse, silo, clouds, trees, etc. is an example of the results that can be achieved.

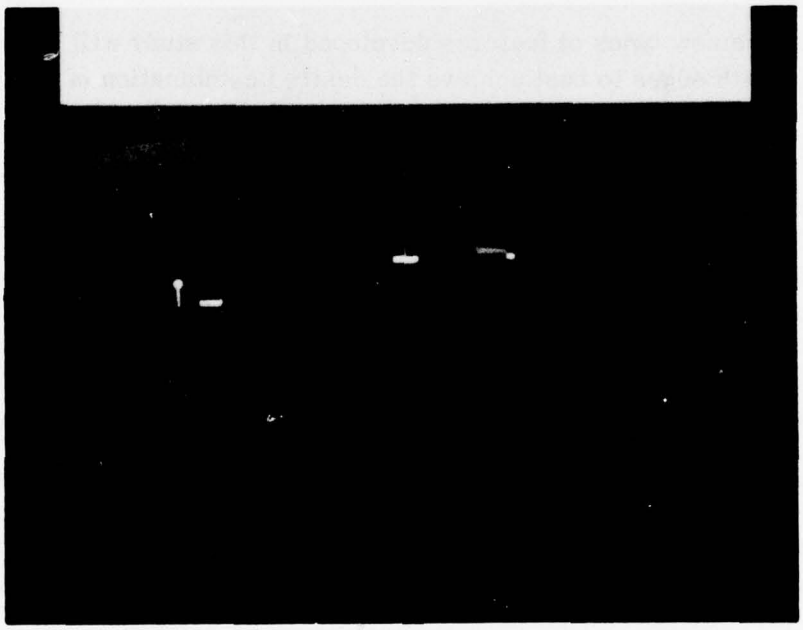
2.3.7 EVALUATION CONCLUSIONS

The circular features lying on the ground are effective in providing surface detail in regions containing few edges. The extensions of the original concept to allow simulation of spheres and the use of the circles for defining specific features, as well as scattered detail, are significant to the goal of achieving maximum training effectiveness in CIG systems.

The scenes included in this section certainly do not represent all methods in which this capability might be applied, but it is believed they suffice to show that elliptical, spherical, and ellipsoidal feature capability would constitute a valuable addition to CIG systems for visual scene simulation in training systems.

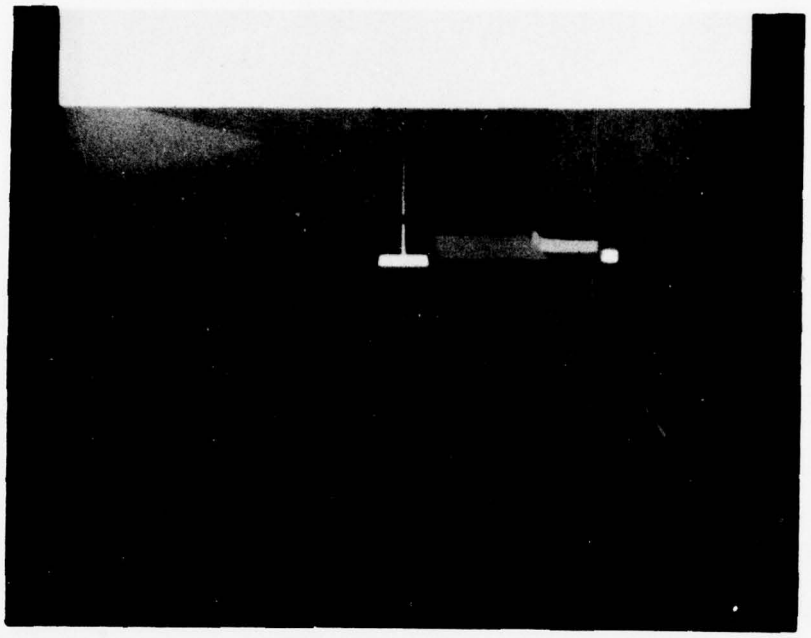


(a)

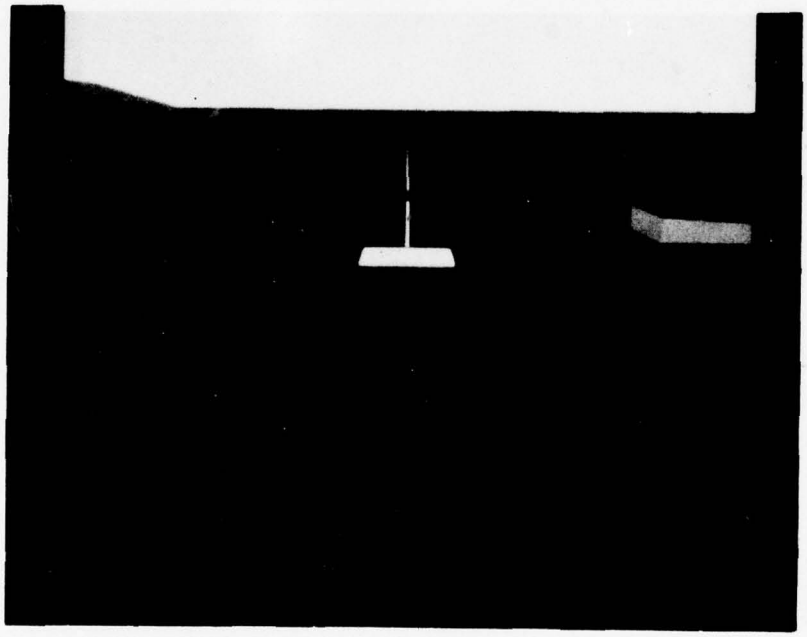


(b)

Figure 8. Runway Scene Sequence (Sheet 1 of 9)

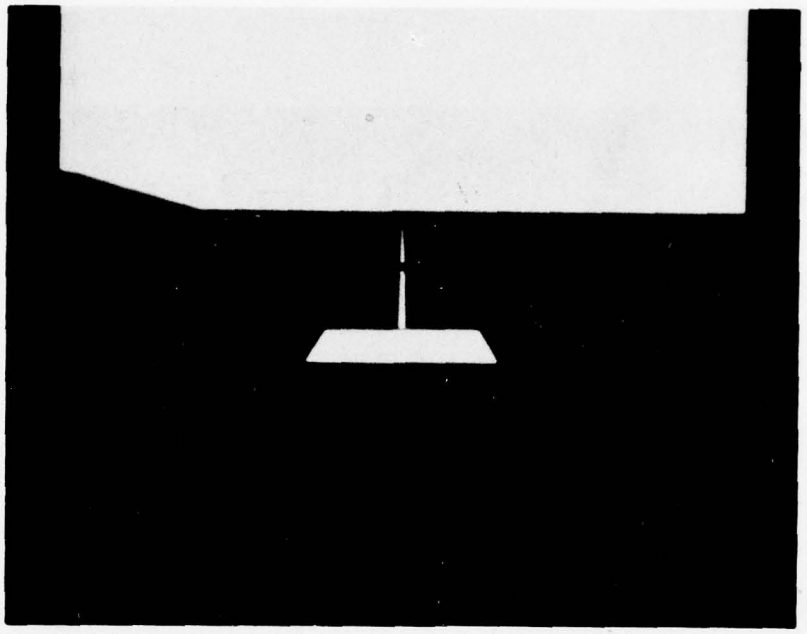


(c)

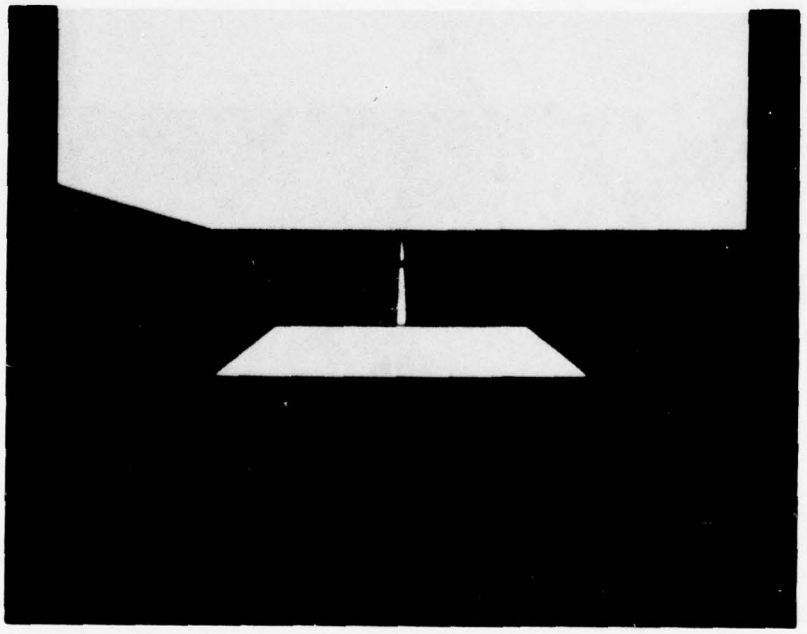


(d)

Figure 8. Runway Scene Sequence (Sheet 2 of 9)

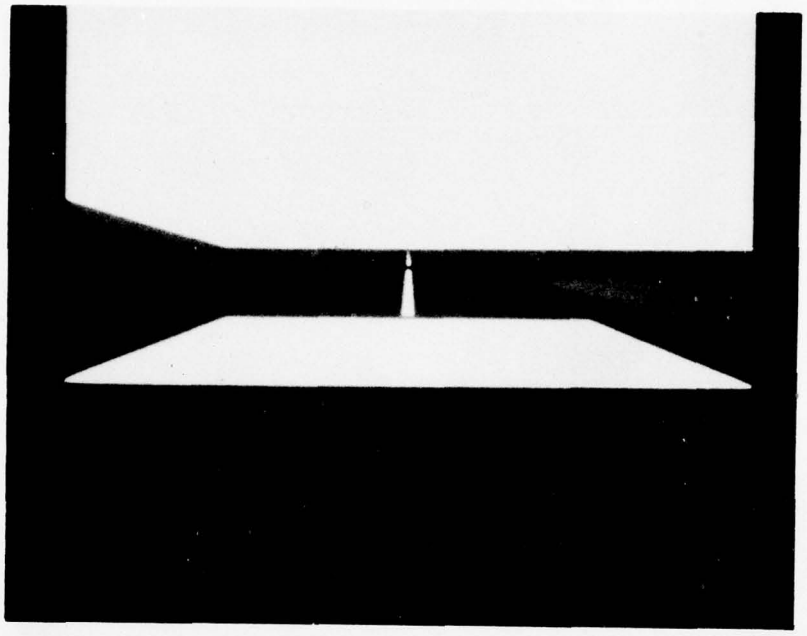


(e)

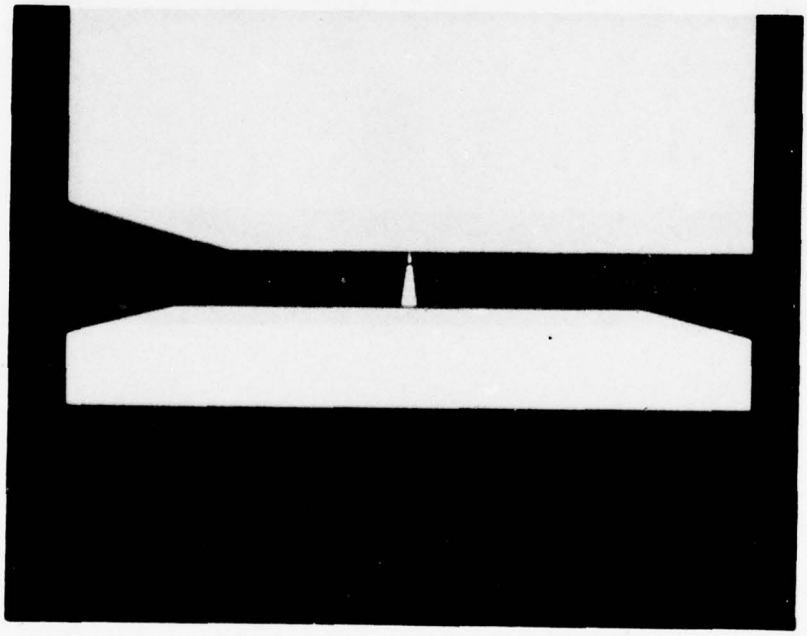


(f)

Figure 8. Runway Scene Sequence (Sheet 3 of 9)

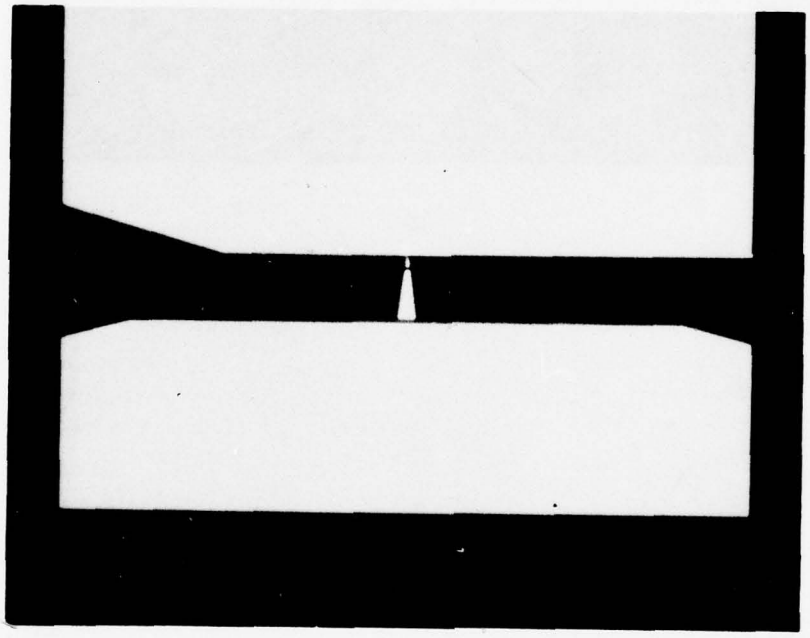


(g)



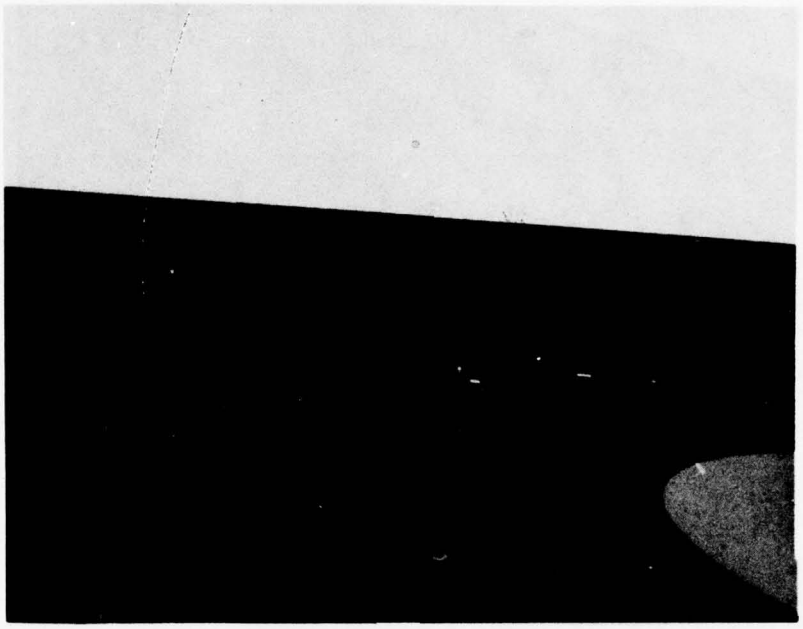
(h)

Figure 8. Runway Scene Sequence (Sheet 4 of 9)

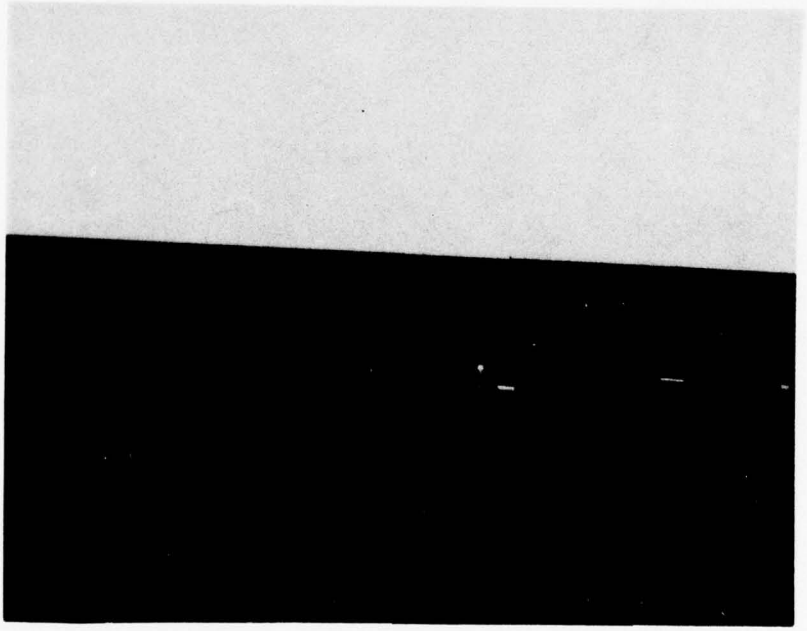


(i)

Figure 8. Runway Scene Sequence (Sheet 5 of 9)

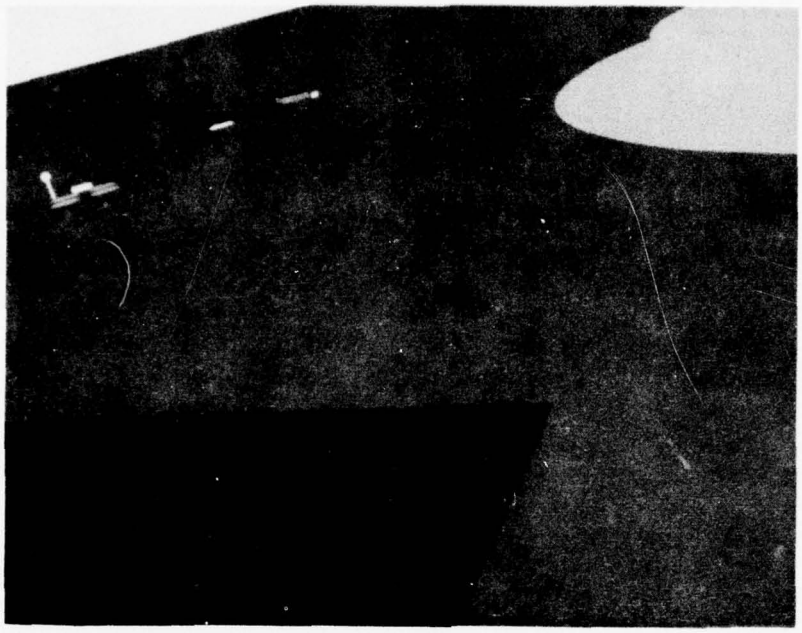


(j)

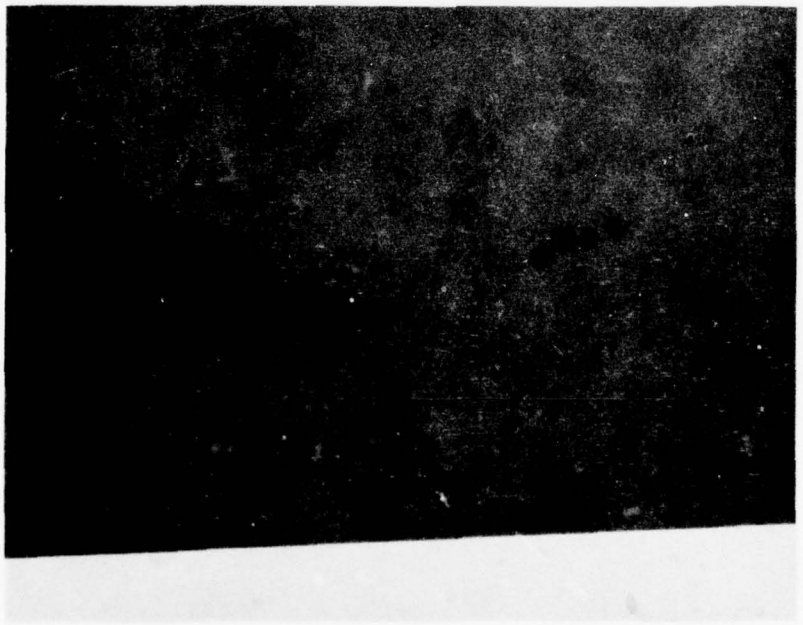


(k)

Figure 8. Runway Scene Sequence (Sheet 6 of 9)

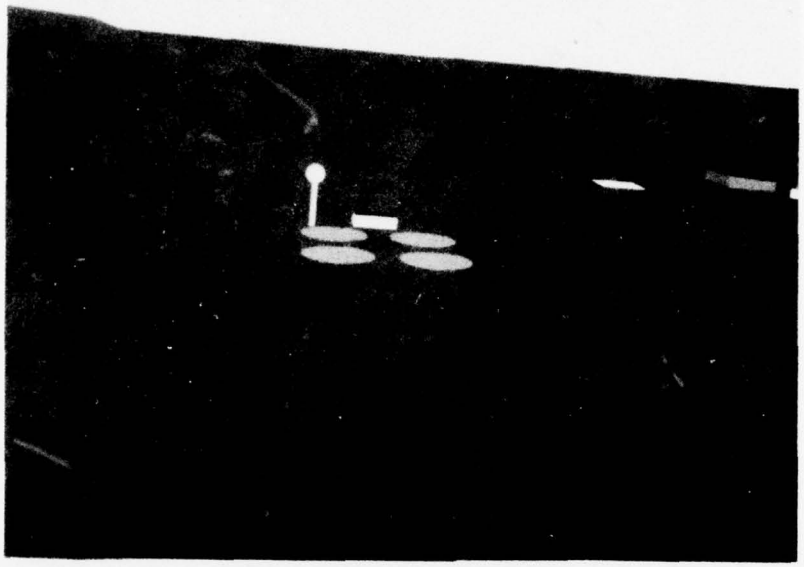


(l)

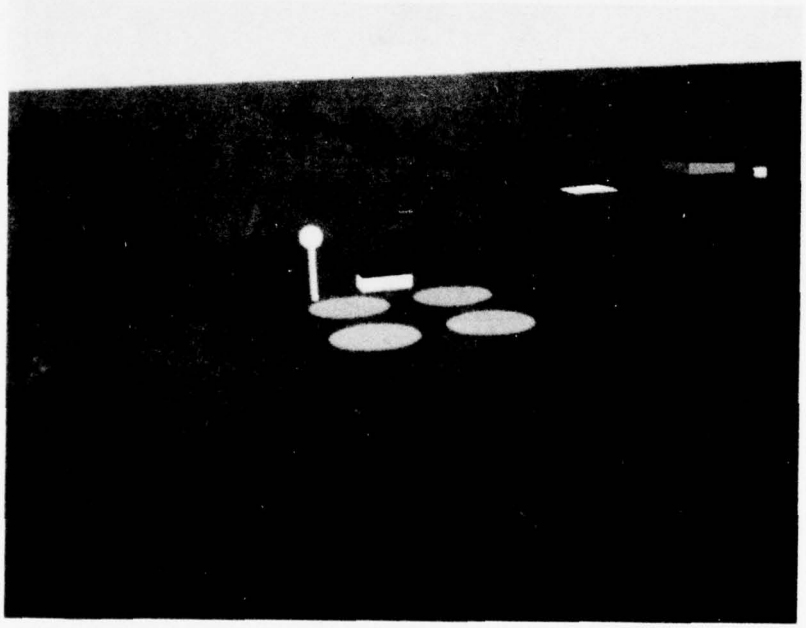


(m)

Figure 8. Runway Scene Sequence (Sheet 7 of 9)

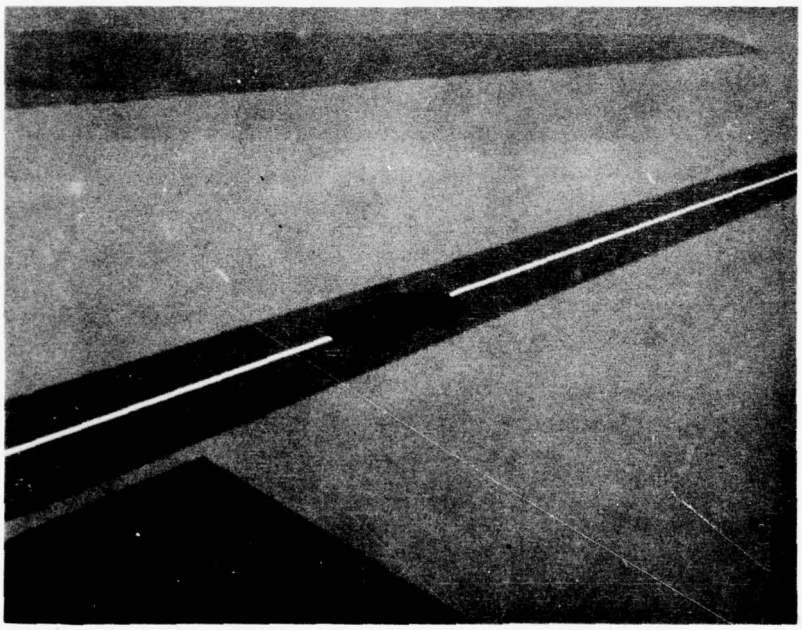


(n)



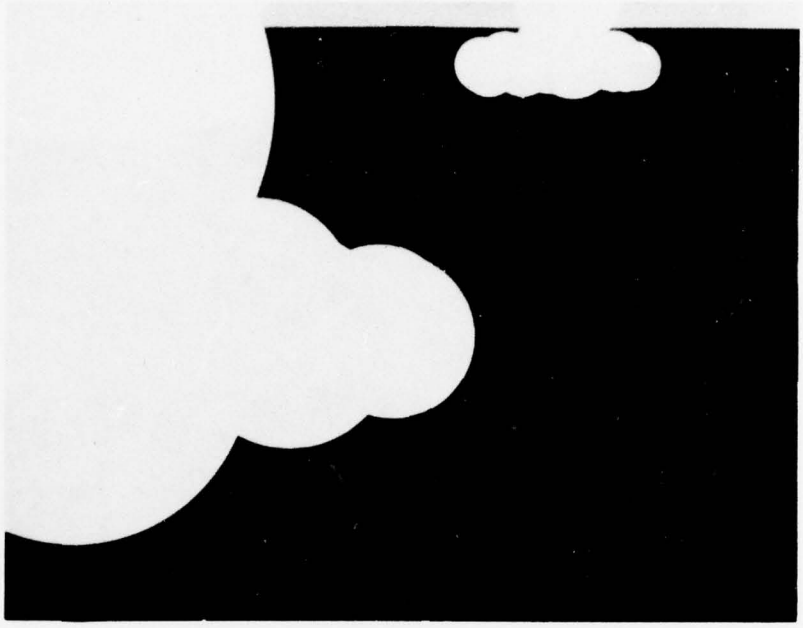
(o)

Figure 8. Runway Scene Sequence (Sheet 8 of 9)

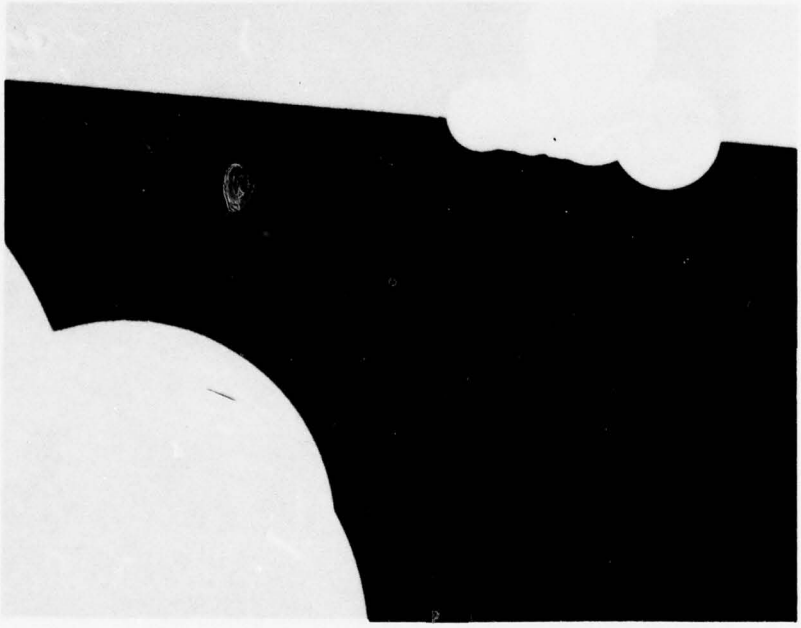


(p)

Figure 8. Runway Scene Sequence (Sheet 9 of 9)

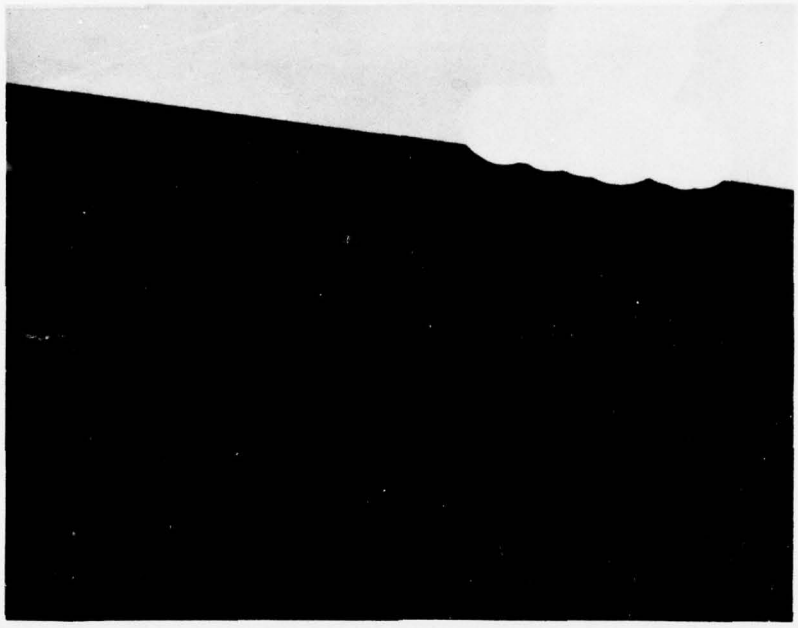


(a)

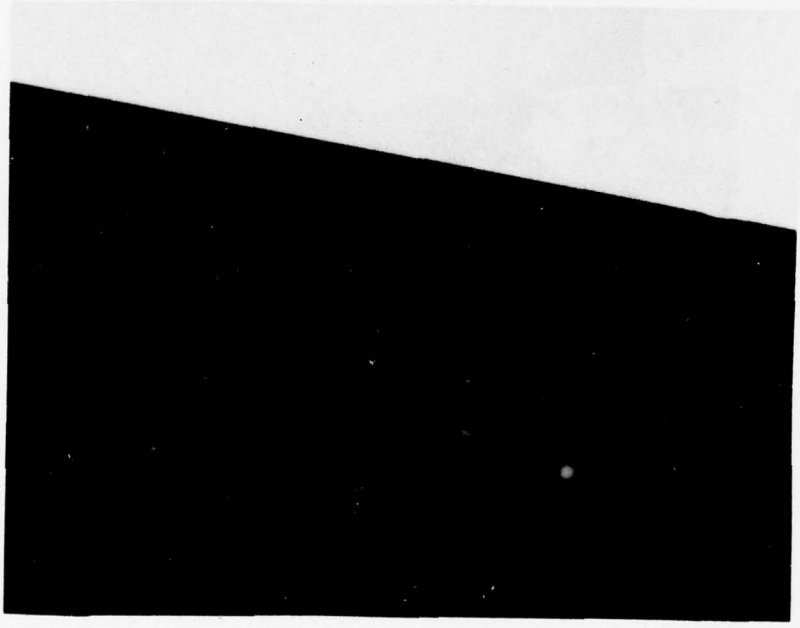


(b)

Figure 9. High-Detail Sequence (Sheet 1 of 3)

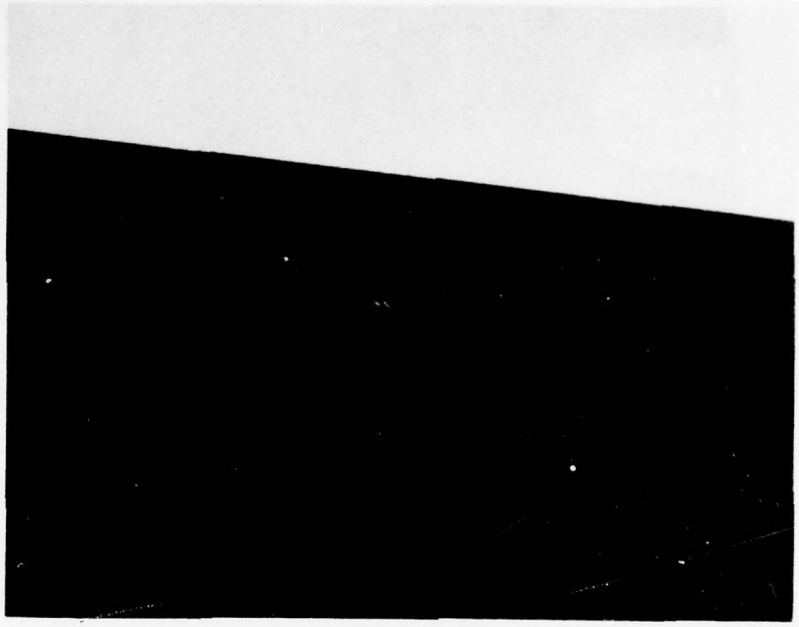


(c)



(d)

Figure 9. High-Detail Sequence (Sheet 2 of 3)



(e)

(f)

Figure 9. High-Detail Sequence (Sheet 3 of 3)

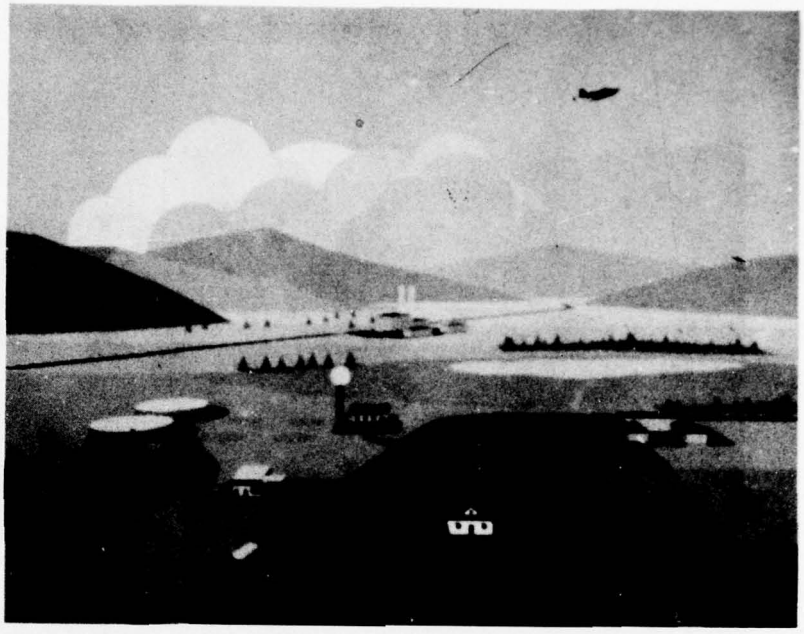


Figure 10. Example Composite Scene

2.4 ALGORITHM DETAILS

The following discussion of algorithm details for simulation of elliptical- and spherical-features is based on the edge algorithm that has been selected as the preferred algorithm for implementation.

2.4.1 CIG ARCHITECTURE

The description of Frame 2 and Frame 3 algorithms involved in simulation of elliptical- and spherical-features becomes more meaningful when considered in the context of a complete CIG system. The relationship is shown on Figure 11 where the functions below the dashed line are specific to the added capability.

Of some significance is the fact that the portion labeled "REST OF FRAME 3" constitutes the bulk of the hardware in an actual system and that this requires no modification for the simulation of circular features. It handles the per-scan-line edges from the circular-feature edge generator just as it does the standard edges.

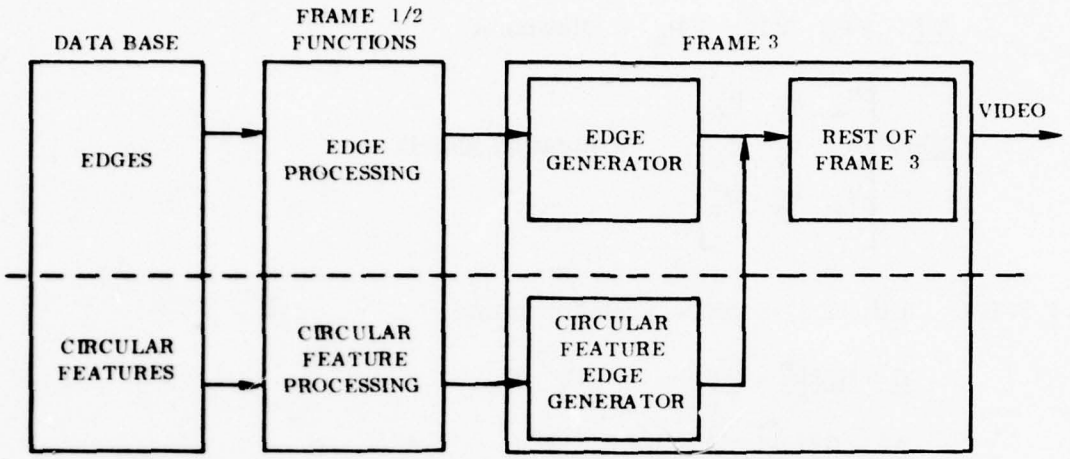


Figure 11. CIG System with Circular Feature Capability

2.4.2 DATA BASE

2.4.2.1 Elliptical Features (EF)

Elliptical Features are spatially defined in the data base by:

Center: $C_x\ C_y\ C_z = \underline{\hat{C}}$

Axis Vector 1: $A1_x\ A1_y\ A1_z = \underline{\hat{A1}}$

Axis Vector 2: $A2_x\ A2_y\ A2_z = \underline{\hat{A2}}$

2.4.2.2 Spherical Features (SF)

Spherical Features are defined in the data base by:

Center $C_x\ C_y\ C_z = \underline{\hat{C}}$

Radius R

2.4.3 FRAME 2 PROCESSING

2.4.3.1 Transform to UVW Space

The following are updated for each new scene, and are used for the spatial transformation from X, Y, Z (data base space) to U, V, W (viewpoint space).

$$\underline{VP} = VP_x \quad VP_y \quad VP_z = \text{Viewpoint}$$

$$\underline{RM} = \begin{bmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ w_x & w_y & w_z \end{bmatrix} = \text{Rotation Matrix}$$

2.4.3.1.1 Elliptical-Feature Transformation

$$\underline{C} = \underline{RM}[\underline{\hat{C}} - \underline{VP}] = C_u \quad C_v \quad C_w$$

$$\underline{A1} = \underline{RM}\,\underline{\hat{A1}} = A1_u \quad A1_v \quad A1_w$$

$$\underline{A2} = \underline{RM}\,\underline{\hat{A2}} = A2_u \quad A2_v \quad A2_w$$

2.4.3.1.2 Spherical-Feature Transformation

$$\underline{C} = \underline{RM}[\underline{\hat{C}} - \underline{VP}] = C_u \quad C_v \quad C_w$$

At this point, we can delete any EF or SF with center behind viewer: If $C_u \leq 0$, Delete.

2.4.3.2 SF to EF Transformation

Each spherical feature is transformed into an elliptical feature valid for the current scene—actually, into a circular feature.

2.4.3.2.1 Determine Increased R

In order for the sphere to be replaced by a circular disc passing through its center, to appear the same size at the eye, the radius of the disc must be greater than that of the sphere. This is illustrated in Figure 12. The new value is determined as follows:

$D^2 = \underline{C} \cdot \underline{C}$. If $D \leq R$, delete SF. Otherwise,

$\text{Sin } \theta = R/D$

$RH = R/\cos \theta$

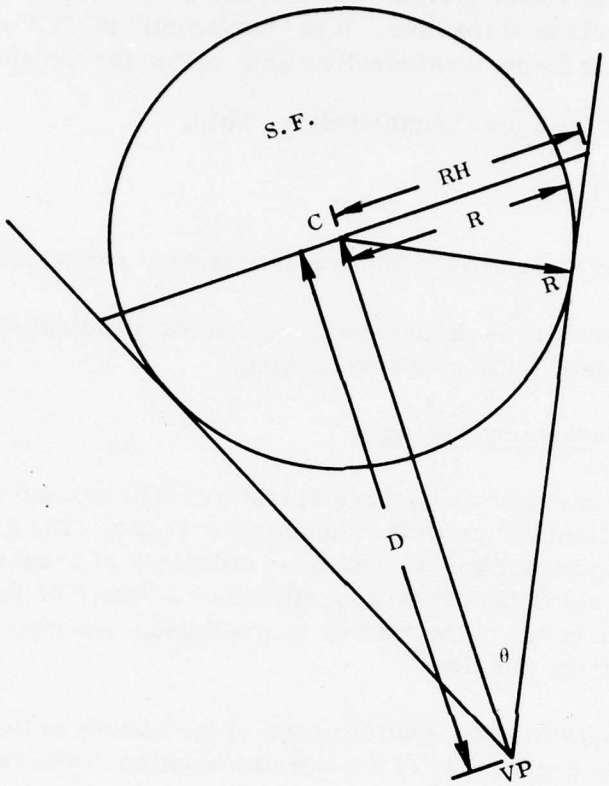


Figure 12. Sphere to Disc Radius Increase

From this, we readily obtain the more hardware-oriented expression:

$$RH = \frac{R\,D}{\sqrt{D^2 - R^2}} .$$

2.4.3.2.2 Axis Vectors for Sphere Replacement

In order that subsequent processing can handle the replacement disc just as any other ellipse, two axis vectors are needed.

$$\underline{V1} = \frac{RH}{\sqrt{C_u^2 + C_v^2}} \begin{bmatrix} -C_v \\ C_u \\ O \end{bmatrix}$$

$\underline{V1}$ is perpendicular to vector $\underline{C}$ from the viewpoint to the center of the sphere; hence, it lies in the plane of the disc. It is "horizontal" in UVW space, it has zero W component. It forms a satisfactory axis vector for the disc.

$\hat{\underline{C}} = \underline{C}/|\underline{C}|$; a unit length vector. Then,

$$\underline{V2} = \underline{V1} \times \hat{\underline{C}}$$

$\underline{V2}$, perpendicular to $\underline{V1}$ and to $\underline{C}$, forms a satisfactory second axis vector.

Now, the spherical feature is defined as an equivalent elliptical feature, and no distinction is made in subsequent processing.

2.4.3.3 Delete Intersecting Features

It is anticipated that circular and spherical features will be used in large numbers to enhance realism and provide valuable visual cues. The goal has, therefore, been to develop algorithms to maximize efficiency of processing such features. A great deal of the processing efficiency is based on the assumption that the view window image of the feature is an ellipse. Among the characteristics which follow from this are:

a. The view window spatial image of the feature is fully defined by the coefficients of a quadratic equation in the variables I and J.

b. Simple processing based only on these coefficients determines the topmost, bottommost, leftmost, and rightmost points of the feature.

c. The topmost and bottommost points obtained above bound the scan lines of the display on which the feature is "active", and on which computations must be performed.

d. Per scan line processing using only the coefficients determines the point on the scan line (the value of J) at which the boundary of a feature intersects the scan line.

The above conditions are violated if a feature intersects the viewplane, the plane containing the view point and normal to the $\underline{u}$ basis vector. This will be clarified by an illustration.

Figure 13 shows the viewpoint at X = 0, Y = 0, Z = 1000. The boresight point is at 0, 1000, 0, and the field of view is as shown. An edge view of an ellipse is shown. The ellipse is partially within the field of view. This fact causes no problem. However, it also crosses the viewplane.

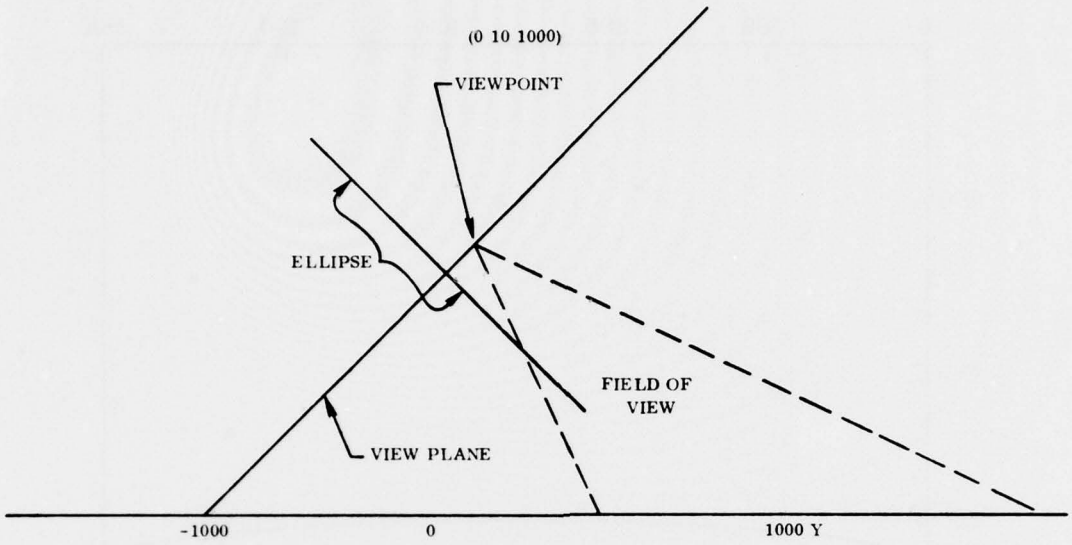


Figure 13. Ellipse Crossing View Plane—Edge View

Standard processing as covered herein gives the set of 6 coefficients defining an image on the view window. It gives I = 86.5, J = 256 as the topmost point, and it gives I = 297.5, J = 256 as the bottommost point. It finds it impossible to achieve a solution for the left and right points.

Figure 14 shows the actual image of the portion of the ellipse visible in the view window. All points on this image satisfy the equation formed by the coefficients found. However, the topmost point on this image is the point identified by the processing as the bottom point.

Figure 15 is an extended view of the viewplane, with the view window boundaries shown in the center. The hyperbola shown is the full figure defined by the coefficients. The reason for the above results is now apparent. It can also be seen that the conditions on which the efficient algorithms were based, are violated. To handle such viewplane-intersecting features, Frame 2 would have to examine the nature of the image, specify to Frame 3 rules for determining the true, "front" image and rejecting the mathematically reflected "back" image. The processing and decision rules by which Frame 3 defines each ellipse in terms of edges for each scan line, would also become more involved.

The present algorithms assume that the model is so designed that, considering the mission to be flown, all features will be entirely outside the view window

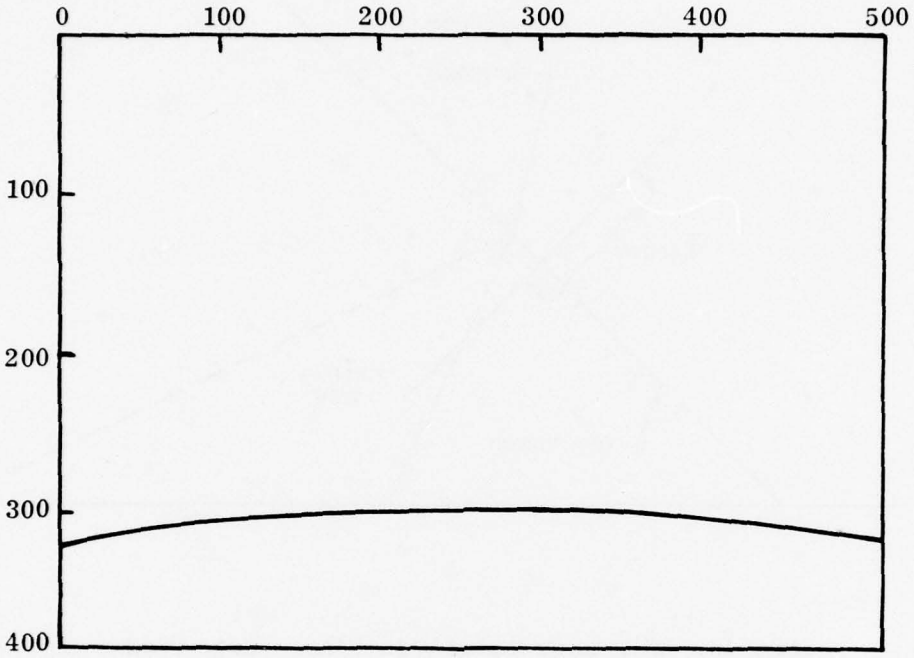


Figure 14. Valid Ellipse Image

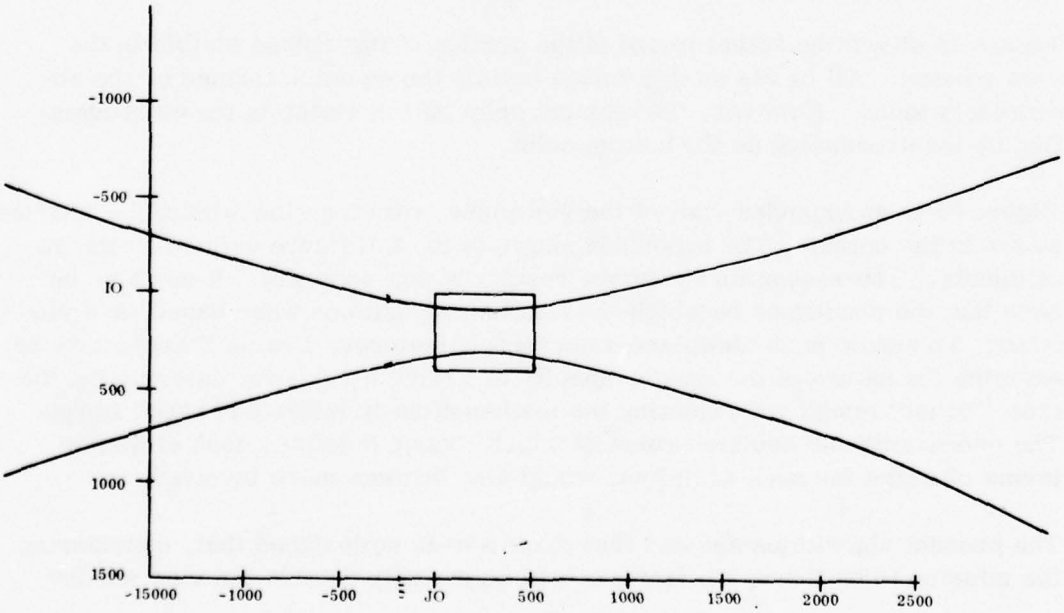


Figure 15. Ellipse Image as Defined by Coefficients

prior to intersecting the viewplane. In the processing, the following step deletes intersecting features:

$$\text{If } (V1_u^2 + V2_u^2) \geq C_u^2; \text{ Delete.}$$

2.4.3.4 Delete End-On Features

If the plane of an elliptical feature contains the view point, its image on the view window is a zero-width straight line. Such features cannot be seen and should be deleted from further processing.

$\underline{N} = \underline{V1} \times \underline{V2}$. $\underline{N}$ is normal to the feature. $|\underline{N}| = R^2$ for circles.

Form a test number: $TST = (\underline{N} \cdot \underline{C}) / (R)\ (\underline{C} \cdot \underline{C})$. TST has the following physical significance: $|TST| \approx \text{Tan } \psi$, where ψ is the minimum angular size of the feature on the display.

For an exactly end-on feature, TST = 0. Comparing it with other values allows deletion of near end-on faces. For example, if we delete the feature when $|TST| < 10^{-3}$ we're deleting when the feature gets smaller than 3.5 arc-minutes.

If we have retained identification of those ellipses which originated as spheres, we can skip the end-on test for them.

Below are some possible alternatives for implementing this test, which may have merit in minimizing hardware requirements.

$R = (|\underline{V1}| + |\underline{V2}|)^{1/2}$. It is invarient—could be stored in data base rather than computed.

We could store $\hat{\underline{N}}_{xyz} = \underline{N}_{xyz} / R$; then $\hat{\underline{N}}_{uvw} = \underline{RM}\ \hat{\underline{N}}_{xyz}$, and

$TST = [\hat{\underline{N}} \cdot \underline{C}] / [\underline{C} \cdot \underline{C}]$.

2.4.3.5 Determine View Window Definition

Now it is necessary to determine the view window definitions of features that have not been deleted prior to this point.

The general equation of an ellipse in view window I-J space is:

$$R1\ I^2 + R2\ J^2 + R3\ I\ J + R4\ I + R5\ J + R6 = 0$$

Frame 2 will determine and send to Frame 3 the coefficients R1 through R6.

In addition, it is efficient to have Frame 2 compute and send to Frame 3 the top, bottom, left, and right extremum points of the ellipse.

2.4.3.5.1 Parameters Defined

NLIN and NELL = total number of lines per scene and elements per line.

I_o, J_o - offset of viewpoint from upper left corner (0, 0) of screen.

VFOV, HFOV—Vertical Field of View, Horizontal Field of View.

C_I = 0.5 NLIN / Tan (0.5 VFOV)

C_J = 0.5 NELL / Tan (0.5 HFOV)

2.4.3.5.2 Coefficient Computation

$$Q1 = \begin{vmatrix} A1_v & A1_u \\ C_v & C_u \end{vmatrix} \qquad Q2 = \begin{vmatrix} A1_u & A1_w \\ C_u & C_w \end{vmatrix} \qquad Q3 = \begin{vmatrix} A1_w & A1_v \\ C_w & C_v \end{vmatrix}$$

$$Q4 = \begin{vmatrix} A2_u & A2_v \\ C_u & C_v \end{vmatrix} \qquad Q5 = \begin{vmatrix} A2_w & A2_u \\ C_w & C_u \end{vmatrix} \qquad Q6 = \begin{vmatrix} A2_v & A2_w \\ C_v & C_w \end{vmatrix}$$

$$Q7 = \begin{vmatrix} A1_u & A1_v \\ A2_u & A2_v \end{vmatrix} \qquad Q8 = \begin{vmatrix} A1_w & A1_u \\ A2_w & A2_u \end{vmatrix} \qquad Q9 = \begin{vmatrix} A1_v & A1_w \\ A2_v & A2_w \end{vmatrix}$$

$$R1H = Q1^2 + Q4^2 - Q7^2$$

$$R2H = Q2^2 + Q5^2 - Q8^2$$

$$R3H = 2\,Q1\,Q2 + 2\,Q4\,Q5 - 2\,Q7\,Q8$$

$$R4H = 2\,Q1\,Q3 + 2\,Q4\,Q6 - 2\,Q7\,Q9$$

$$R5H = 2\,Q2\,Q3 + 2\,Q5\,Q6 - 2\,Q8\,Q9$$

$$R6H = Q3^2 + Q6^2 - Q9^2$$

The above define the ellipse in a form independent of field of view, number of lines, or number of elements. Define I' = (tangent of vertical component of angle between a ray from the viewpoint and a normal to the view window). Define J'' in the same manner for the horizontal component of the angle. The ellipse is now:

$$R1H\,I'^2 + R2H\,J'^2 + R3H\,I'\,J'\,R4H\,I' + R5H\,J' + R6H = 0$$

Now consider the actual line and element numbers on the view window:

$I = C_I\,I' + I_o$. $J = C_J\,J' + J_o$. Substituting, we can obtain the coefficients for the ellipse in terms of I and J:

$$R1 = R1H/C_I^2$$

$$R2 = R2H/C_J^2$$

$$R3 = R3H/(C_I\,C_J)$$

$$R4 = R4H/C_I - (2\,R1H\,I_o)/(C_I^2) - (R3H\,J_o)/(C_I\,C_J)$$

$$R5 = R5H/C_J - (2\,R2H\,J_o)/(C_J^2) - (R3H\,I_o)/(C_I\,C_J)$$

$$R6 = \frac{R1H\,I_o^2}{C_I^2} + \frac{R2H\,J_o^2}{C_J^2} + \frac{R3H\,I_o\,J_o}{C_I\,C_J} - \frac{R4H\,I_o}{C_I} - \frac{R5H\,J_o}{C_J} + R6H$$

The above coefficients, R1 through R6, spatially define the view window image of the elliptical or spherical feature being simulated.

2.4.3.5.3 Extremum Point Computation

2.4.3.5.3.1 Top and Bottom. The equation of the ellipse can be solved for J to give:

$$J = \frac{-(R3I+R5) \pm \sqrt{(R3I+R5)^2 - 4\,R2\,(R1I^2 + R4I + R6)}}{2\,R2}$$

J will, in general, be a double-valued function of I, except at the top and bottom points of the ellipse where the above expression will be single-valued. These will be at the values of I making the discriminant equal to zero. These values of I are found by solving the resulting quadratic equation:

$$I^2 [R3^2 - 4\ R1\ R2] + I [2\ R3\ R5 - 4\ R2\ R4] + [R5^2 - 4\ R2\ R6] = 0$$

The lesser of the two solutions of this equation is the value of I at the top, the greater at the bottom. For each, solve for J from:

$$J = - \frac{R3\ I + R5}{2\ R2}$$

2.4.3.5.3.2 Left and Right. Proceeding as above, we get the following to solve for J:

$$J^2 [R3^2 - 4\ R1\ R2] + J [2\ R3\ R4 - 4\ R1\ R5] + [R4^2 - 4\ R1\ R6] = 0$$

Then, obtain I from:

$$I = - \frac{R3\ J + R4}{2\ R1}$$

2.4.3.6 Frame 2 Conclusion

This concludes the Frame 2 algorithms for definition of elliptical, circular, and spherical features. Frame 2 sends to Frame 3 the six coefficients, R1 through R6, the four extremum points, and the tone and priority for each feature.

2.4.4 FRAME 3 PROCESSING

A given feature may have been originally defined as a circle or an ellipse of arbitrary location and orientation, or as a sphere. Whatever the source, it is defined to Frame 3 as an ellipse in the I-J view window space. The six coefficients, R1 through R6 of the following equation are given:

$$R1\ I^2 + R2\ J^2 + R3\ I\ J + R4\ I + R5\ J + R6 = 0$$

In addition, the extremum points:

$$\begin{bmatrix} I_T \\ J_T \end{bmatrix} \begin{bmatrix} I_B \\ J_B \end{bmatrix} \begin{bmatrix} I_L \\ J_L \end{bmatrix} \begin{bmatrix} I_R \\ J_R \end{bmatrix}$$

are precomputed by Frame 2 and provided to Frame 3.

The Frame 3 circular-feature edge generator performs the following functions for each scan line generated.

2.4.4.1 Delete Non-Active Features

A simple test deletes features for any scan line to which they do not contribute. When computing scan line LIN, LIN quantitatively defines the top of the scan line, (LIN+1) the bottom. For each feature:

If $I_T \geq (LIN+1)$ or $I_B \leq LIN$, go to next feature.

2.4.4.2 Define Critical Points

Eight points are defined for each feature, as illustrated on Figure 16. Two arrays are generated, KEPT(N), N = 1, 8; and RJPT(N), N = 1, 8. These are the applicability and element number for each of the eight points. As designated below, the drawing on Figure 16, KEPT for an intersection is set to 1 if the intersection exists. For extrema, KEPT is set to 1 if the point is within the scan line. Next, we will use this information to define the edges that define the feature for the current scan line.

Values of I for the various points are either extremum values, [RIL(NF), for example], the value of I at the top of the current scan line (RLIN), or the value of I at the bottom of the current scan line (RLP1).

Values of J are either extremum values, [RJL(NF)], or computed intersection values, [RJPT(2)].

For points 1, 3, 5, and 7, the value of I provided by Frame 2 is compared with LIN and LIN+1 to determine KEPT for the points.

For the intersections, proceed as follows. J can be solved in terms of I as:

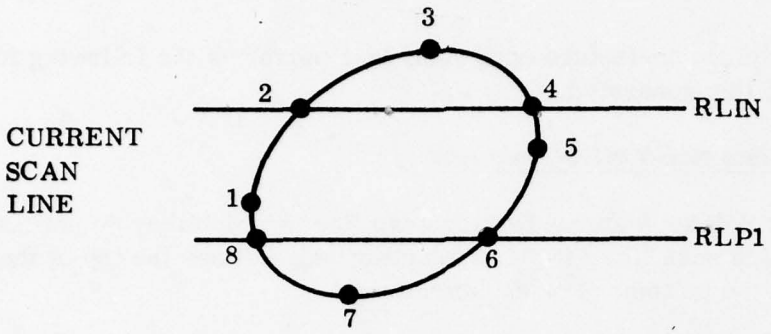
$$J = \frac{-(R3I+R5) \pm \sqrt{(R3I+R5)^2 - 4\ R2\ (R1\ I^2 + R4\ I + R6)}}{2\ R2}$$

Use RLIN for I in the above. If the discriminant is zero or negative, KEPT(2) and KEPT(4) are zero; otherwise they are 1, and the above is used to solve for RJPT(2) and RJPT(4).

Using (RLIN+1), obtain the same information for points 6 and 8.

2.4.4.3 Form Edges

Now, we have all needed information on the eight points. We are ready to form edges. Figure 17 has been prepared to help clarify the process. If we consider



POINT NO.	I	J	KEPT NO.
1	RIL(NF)	RJL(NF)	1 if within scan line
2	RLIN	RJPT(2)	1 if intersection exists
3	RIT(NF)	RJT(NF)	1 if within scan line
4	RLIN	RJPT(4)	1 if intersection exists
5	RIR(NF)	RJF(NF)	1 if within scan line
6	RLP1	RJPT(6)	1 if intersection exists
7	RIB(NF)	RJB(NF)	1 if within scan line
8	RLP1	RJPT(8)	1 if intersection exists

Figure 16. Ellipse Point Definitions

PER SCAN LINE ELLIPSE DEFINITION

CASE	KEPT ( )			
	1	5	2	6
1	0	0	0	0
2	0	0	0	1
3	0	0	1	0
4	0	0	1	1
5	0	1	0	0
6	0	1	0	1
7	0	1	1	0
8	0	1	1	1

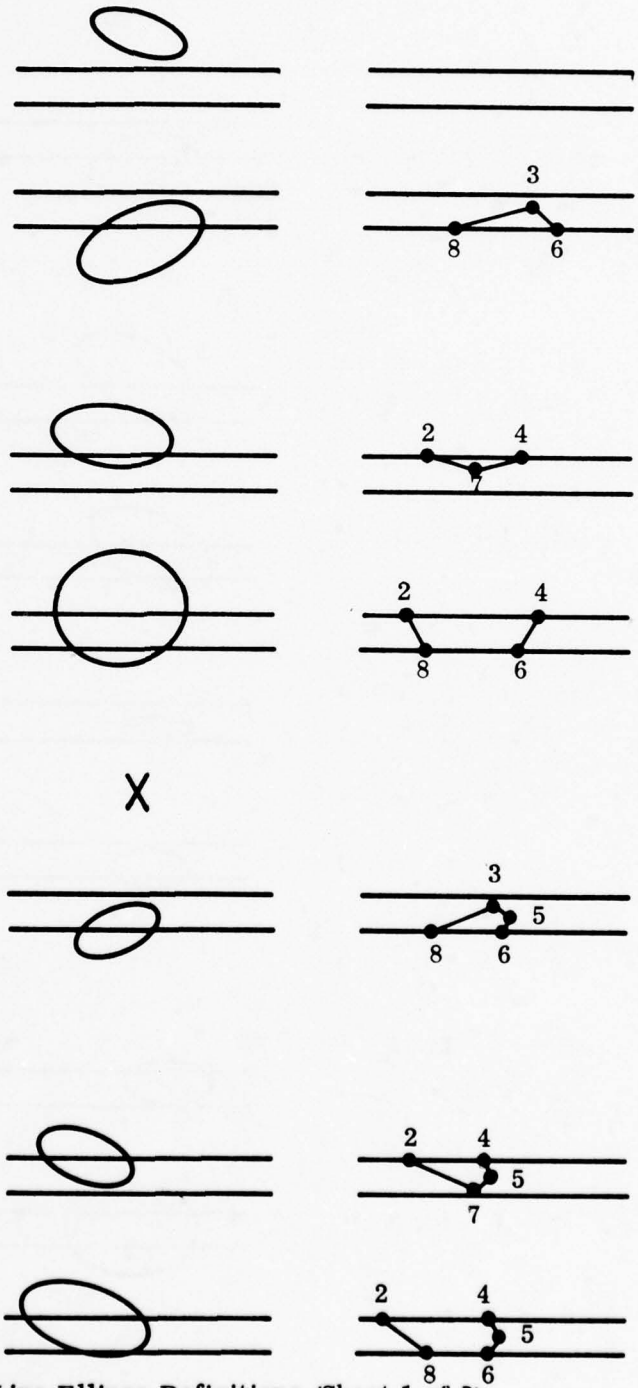


Figure 17. Per Scan Line Ellipse Definitions (Sheet 1 of 2)

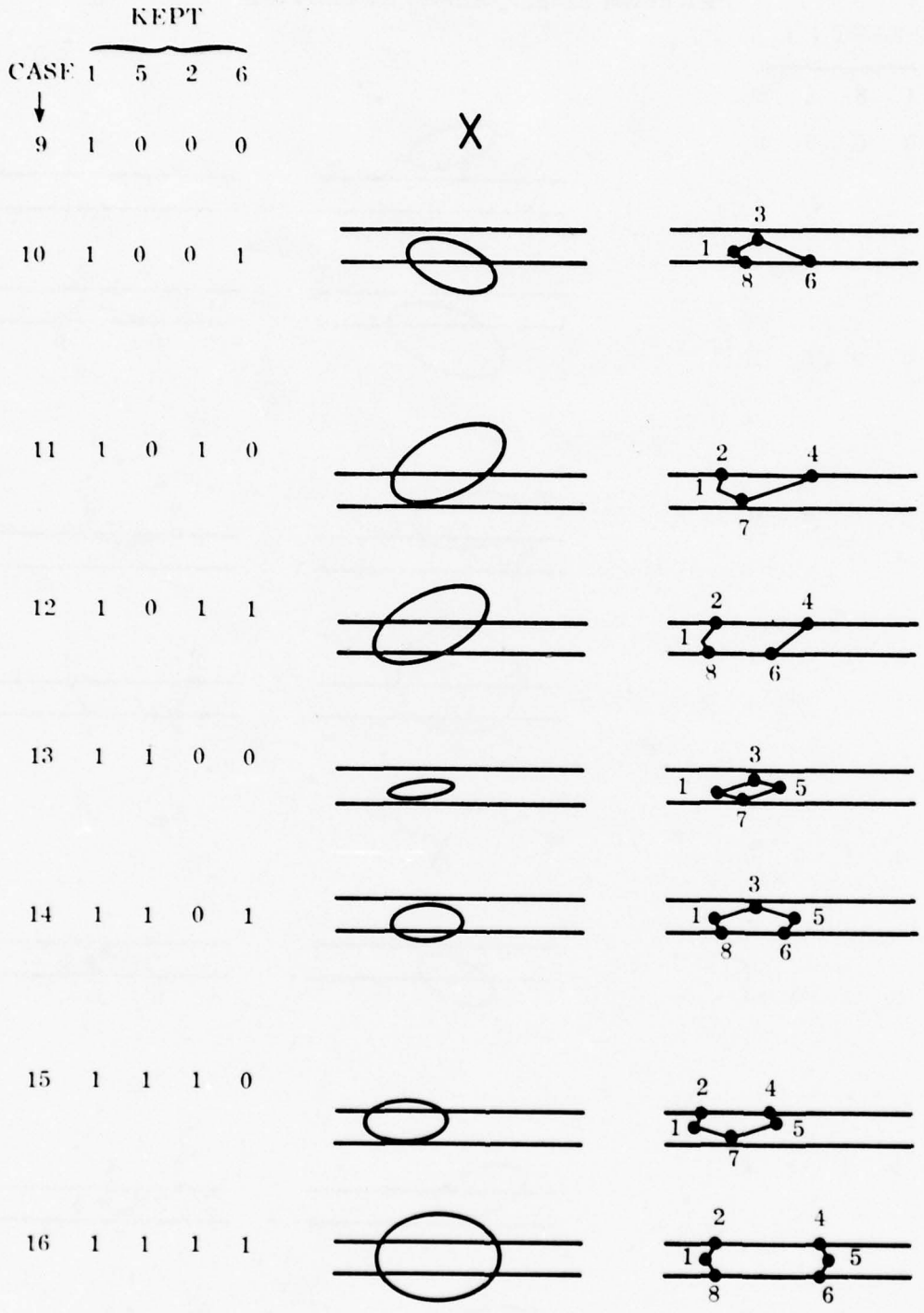


Figure 17. Per Scan Line Ellipse Definitions (Sheet 2 of 2)

the binary number formed by KEPT(1), KEPT(5), KEPT(2), and KEPT(6), we obtain a numbering for 16 possible cases representing relationships between an ellipse and a scan line. Two are not physically realizable. The others are illustrated, with the actual ellipse shown, and the edges representing it on the current scan line.

The algorithm, used by Frame 3 to determine the edges for the cases as shown, is as follows. Consider 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 1 - 2 - 3 as representing a circular ring of the numbers 1 through 8. Determine from this the sequence of point numbers which have KEPT equal to 1, continuing until a point is repeated. These points are the vertices of a set of potential edges. Deleting potential edges coincident with the top or bottom of the scan line, leaves the actual edges.

The non-trivial cases are covered in Table 2.

Table 2

Non-Trivial Point Sequences

Case	Sequence	Potential Edges	Actual Edges
2	3 6 8 3	3	2
3	2 4 7 2	3	2
4	2 4 6 8 2	4	2
6	3 5 6 8 3	4	3
7	2 4 5 7 2	4	3
8	2 4 5 6 8 2	5	3
10	1 3 6 8 1	4	3
11	1 2 4 7 1	4	3
12	1 2 4 6 8 1	5	3
13	1 3 5 7 1	4	4
14	1 3 5 6 8 1	5	4
15	1 2 4 5 7 1	5	4
16	1 2 4 5 6 8 1	6	4

The non-edges among the potential edges may be detected by the fact that their $I_1 = I_2$, or by excluding the 2 through 4 and 6 through 8 sequences.

The edges are now ordered, have their priority and color information appended, and are then sent to the remainder of Frame 3 processing where they are indistinguishable from edges derived from faces or from defined point lights.

This completes the algorithms for elliptical- and spherical-feature generation using the selected edge algorithm.

2.5 HARDWARE ESTIMATE SUMMARY

Following is a preliminary estimate of the hardware requirements for real-time implementation of circular-feature capability.

The circular-feature implementation will consist of two functions, a Frame 2 circular-feature processor and a Frame 3 circular-feature edge generator. Since normal vertex processing requires rotation from reference (X, Y, Z) coordinates to viewpoint (U, V, W) coordinates, the same equipment would be used to rotate circular feature centroid and axis vectors from reference to viewpoint coordinates. The circular-feature processor will then compute the extreme points and coefficients for each potentially visible circular feature. This processor will require approximately 120 logic boards to perform these computations plus 7 output buffer memory boards for each 256 potentially visible circular features.

The Frame 3 circular-feature edge generator function will compute the per-scan-line edges for all circular features intercepting the line. On this basis, the Frame 3 processing capacity, following the circular-feature edge generator, is limited to 256 edges per line. The circular-feature edge generator was sized to process up to 64 active circular features per line. This permits using a single computational path for the associated multiple edges. The computations will require 45 logic boards, plus 10 semiconductor memory boards for each 256 potentially visible circular features. For example, a system capability of 1024 potentially visible features will require 45 logic boards, plus 40 memory boards.

NOTE

In the above, "board" refers to a standard Compu Scene board, 5 inches by 7 inches, capable of holding 40 integrated circuits.

2.6 HARDWARE ESTIMATE DETAILS

The elliptical-feature implementation will consist of two functions, a Frame 2 elliptical-feature processor and a Frame 3 elliptical-feature edge generator.

2.6.1 FRAME 2 FUNCTIONS

Figure 18 provides a functional block diagram of Frame 2 modified to include elliptical-feature processing. The significant difference between a basic Frame 2 and the one shown in Figure 18 is the addition of an elliptical-feature processor.

Elliptical features are spatially defined in the data base by:

Center:	C_x	C_y	C_z	$= \hat{\underline{C}}$
Axis Vector 1:	$A1_x$	$A1_y$	$A1_z$	$= \hat{\underline{A1}}$
Axis Vector 2:	$A2_x$	$A2_y$	$A2_z$	$= \hat{\underline{A2}}$

These data forms are compatible with existing environment store formats and, therefore, will not significantly affect this function. Initial processing of the elliptical features performs conversion from environment (data base) coordinates to viewpoint coordinates. This yields the elliptical features in the following form:

$$\underline{C} = \underline{RM}[\hat{\underline{C}} - \underline{VP}] = C_u \quad C_v \quad C_w$$

$$\underline{A1} = \underline{RM}\,\hat{\underline{A1}} = A1_u \quad A1_v \quad A1_w$$

$$\underline{A2} = \underline{RM}\,\hat{\underline{A2}} = A2_u \quad A2_v \quad A2_w$$

Since normal vertex processing requires rotation from reference (X, Y, Z) coordinates to viewpoint (U, V, W) coordinates, the same equipment would be used to rotate circular-feature centroid and axis vectors from reference to viewpoint coordinates. This process is performed by the vertex processor function in Figure 18. The next computation in the elliptical-feature processing requires the information of nine "Q" terms for each feature. These are defined in paragraph 2.4.3.5.2.

The computations required to obtain these terms are similar to existing computations in the Frame 2 face-edge processor. It is assumed these can be utilized to compute the Q's with minor control modification. As shown in Figure 18, the elliptical-feature Q's and associated data will be sent to the dedicated elliptical-feature processor.

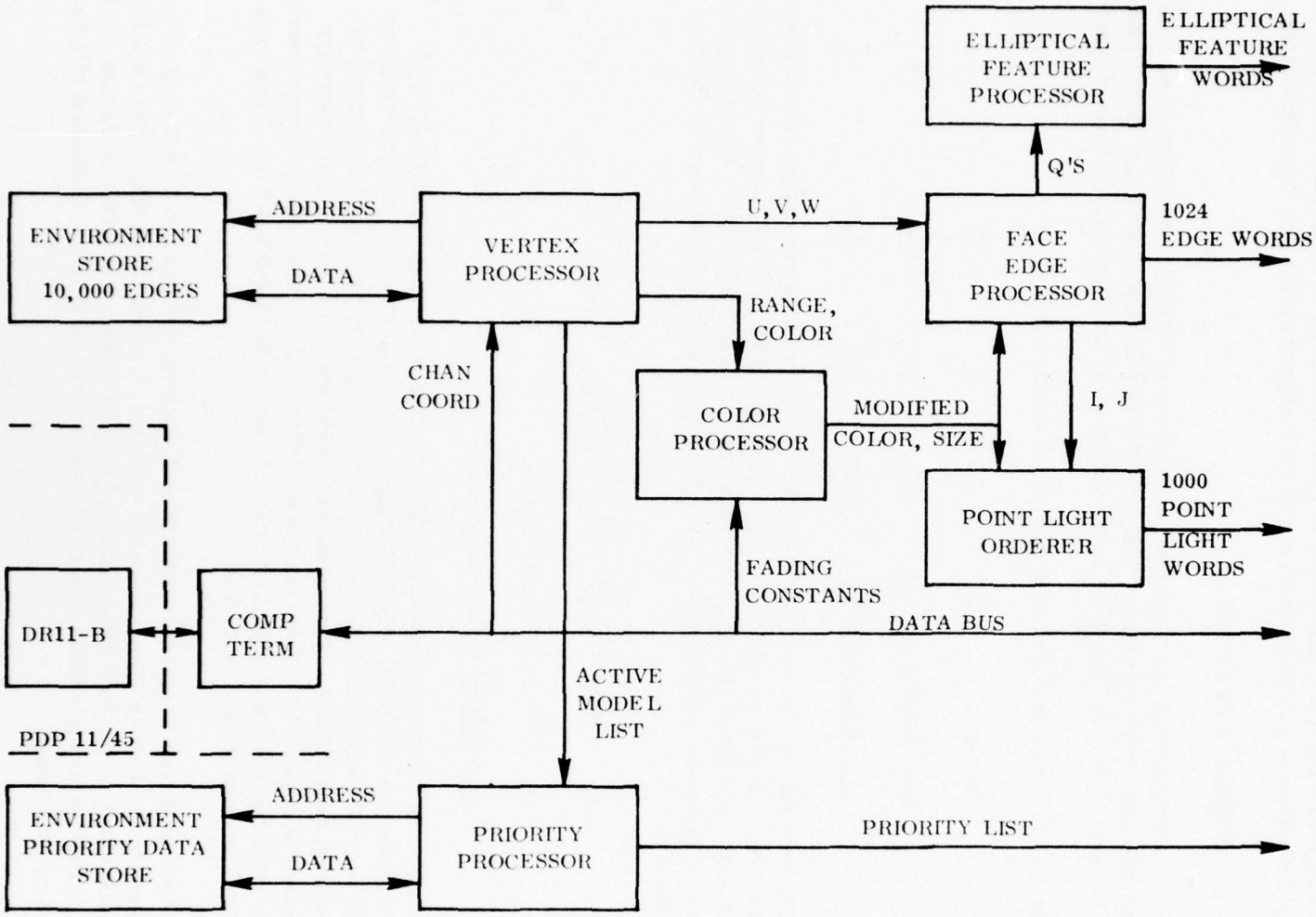


Figure 18. Frame 2 Special-Purpose Computer Functional Block Diagram

The elliptical-feature processor will compute the ellipse extremum points and coefficients required by the Frame 3 elliptical-feature edge processor for line-edge computations. To estimate the hardware complexity of the Frame 2 elliptical-feature processor, the computation of the ellipse extreme points was diagrammed as shown in Figure 19. The Q's will be received-time sequenced as three groups of three Q's as shown at the lower left of Figure 19. At the points noted in the diagram, the R(N)H terms will be obtained as an intermediate step in the pipeline process. Then these terms will be used to compute the four ellipse extremes of top, bottom, left, and right. The four extremes will be computed in time sequence so that the same computational hardware paths can be used for these terms. Basic data processing speed would be one extreme point per microsecond, which is equivalent to four microseconds per elliptical feature. This is equivalent to the process rate for the edges of a triangular face.

The computational elements shown in Figure 19 correspond to standard logic board designs utilized in the present Frame 2 functions. The application of these board types to Figure 19 created a total board count for this computation. This total was weighted by an additional experience factor accounting for timing, control, and self-test overhead to obtain a final estimate of 120 logic boards to perform the elliptical-feature processor computations. This must be supplemented by seven output buffer memory boards for each 256 potentially visible elliptical features. As an example, 28 output memory boards would be required in a system intended to have a capacity of 1000 potentially visible elliptical features.

At this point, it should be noted that the implementation of elliptical features in conjunction with the indicated Frame 2 configuration is not an optimum system design. The computational rate of the elliptical-feature processor must be high in order to be compatible with the processing of face edges and point lights. However, its duty cycle or utilization rate will be relatively low for the large amount of dedicated hardware. Consequently, it is not the most cost effective implementation. To improve this situation, the Frame 2 total set of processes should be resturctured for a more efficient implementation. This is beyond the scope of this study.

2.6.2 FRAME 3 FUNCTIONS

The Frame 3 elliptical-feature edge generator function, see Figure 20, will compute the per-scan-line edges for all elliptical features intercepting a line. The elliptical-feature data for a scene will be transferred from the Frame 2 processor to the Frame 3 edge generator at the start of the raster frame period. Figure 21 provides a functional block diagram of the elliptical-feature edge generator which is identical to a standard edge generator except for the line boundary calculation and edge select functions. The feature scanners operate on 256 features per scanner to pre-select the active features for each

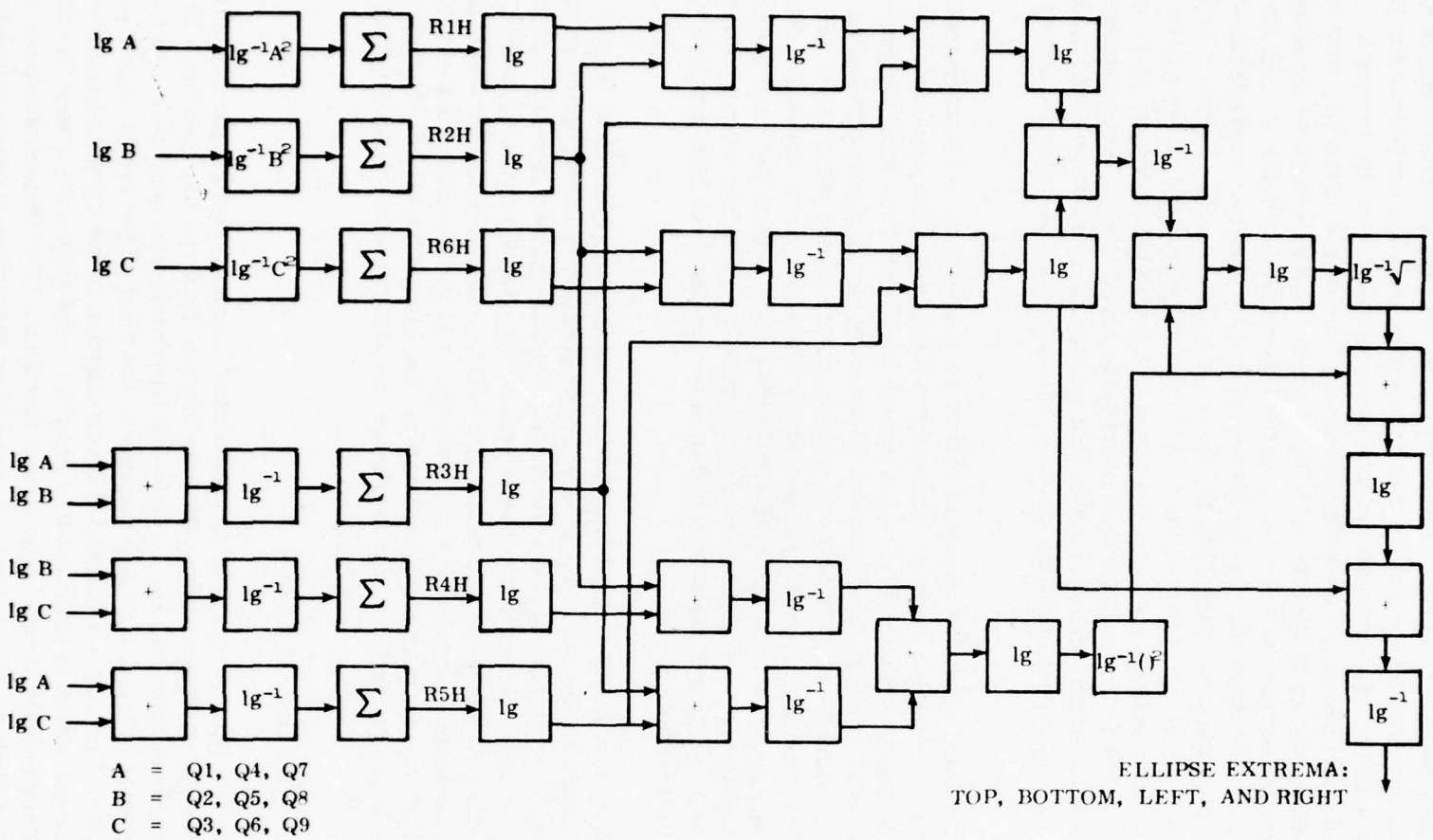


Figure 19. Computation of Ellipse Extreme Points

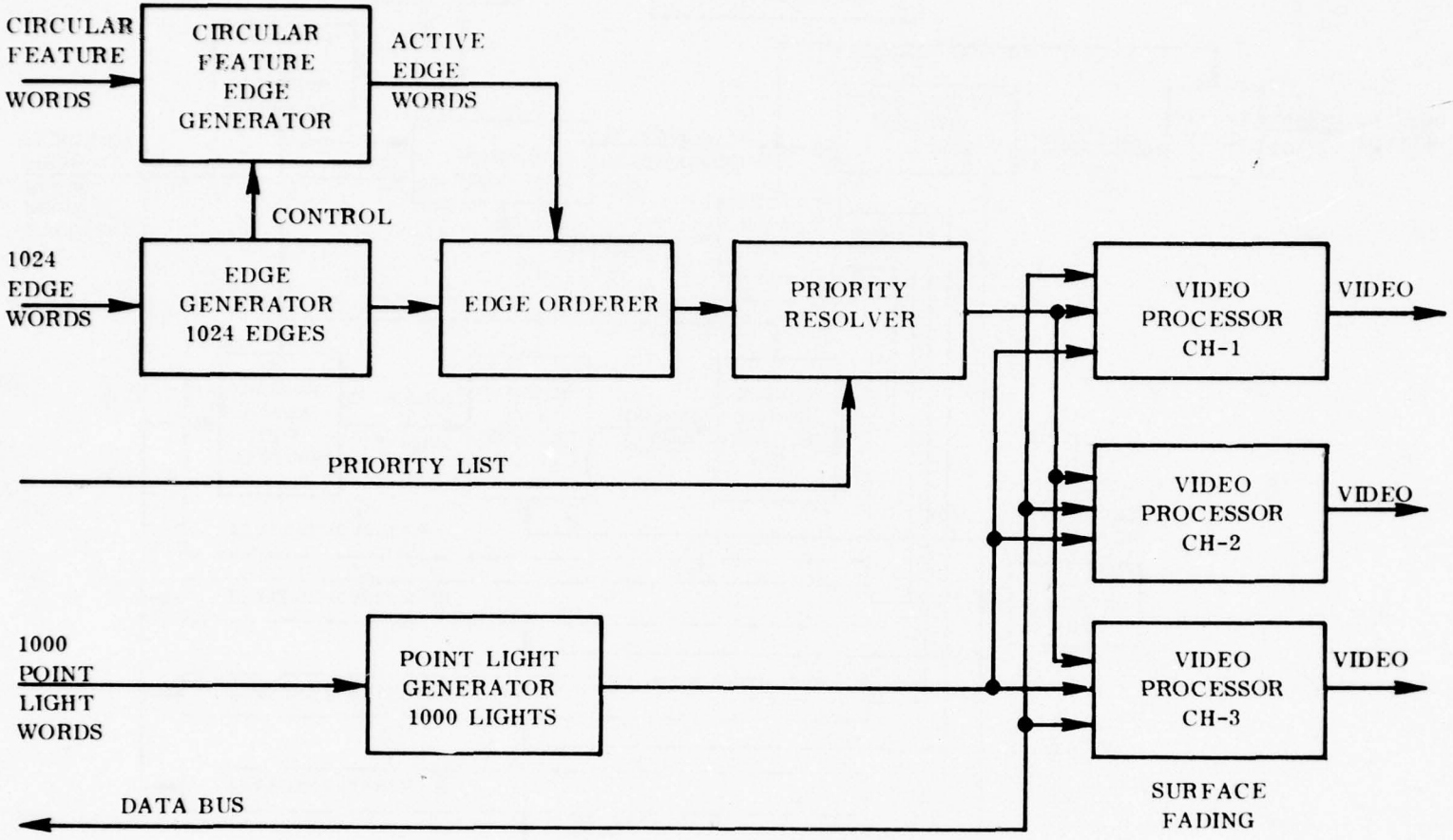


Figure 20. Frame 3 Special-Purpose Computer Functional Block Diagram

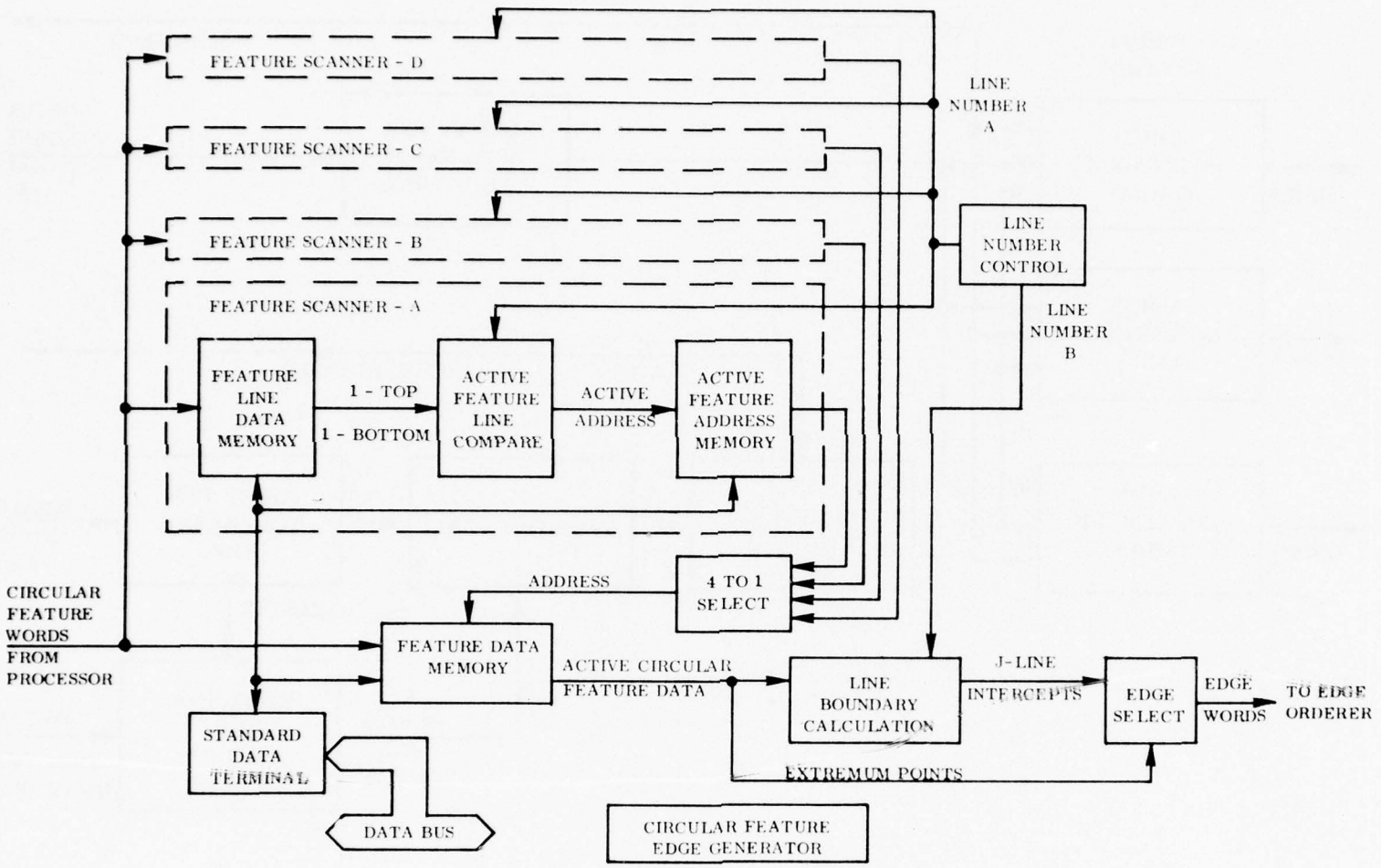
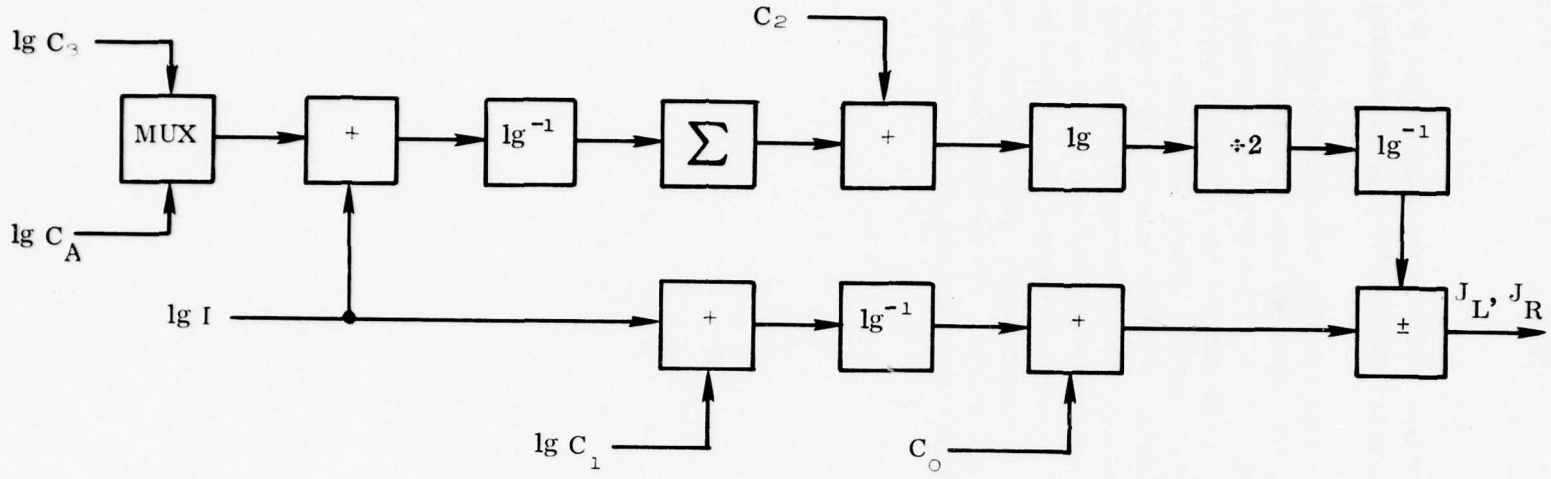


Figure 21. Circular-Feature Edge Generator

raster line in the form of an address list. This active address list is then used to access the feature data from temporary storage for edge processing.

On the basis that the Frame 3 processing capacity following the elliptical-feature edge generator is limited to 256 edges per line, the circular-feature, edge generator was sized to process up to 64 active elliptical features per line. This permits using a single computational path for the multiple edges (four maximum) associated with an elliptical feature. The line boundary computation is shown in elemental form in Figure 22. Standard logic board designs were used to estimate a total for this function. This estimate was then used to adjust the standard edge generator board complement in order to obtain an elliptical-feature edge generator estimate. The elliptical-feature edge generator will require a basic set of 45 logic boards to which must be added 10 logic boards for each 256 potentially visible elliptical features. The latter accounts for adjusting the number of feature scanners and the size of the feature data memory to provide a given feature capacity.



$$J = C_0 + C_1 I \pm \sqrt{C_4 I^2 + C_3 I + C_2}$$

Figure 22. Line Boundary Computation

SECTION 3

CONTOURS

3.1 APPROACH

A contour may be defined as a nonflat surface feature. Some descriptive terms that come to mind are ridges and valleys, hills and dales. Contours in large numbers can obviously provide the surface detail needed for motion cues. In addition, the changing relation between a three-dimensional contour and the horizon provides far more sensitive cues to slight altitude changes in low-level flight than features defined entirely on the surface.

Actual ground contours can be approximated by edges and faces, and scenes can be generated by standard CIG systems. Figure 23, produced during an earlier study contract, is an example of this. This approach is, however, rather expensive in edge usage.

Another approach is to generate a tonal variation on what is spatially defined as a planar surface to give the impression of vertical contouring. Figure 24, made during an IR&D investigation, is an illustration of this. Any CIG system that includes the gradient algorithm used for curvature simulation can apply it in this manner. It can, subject to certain constraints, give a realistic simulation of smooth, low-amplitude, contour variations. It can cover a given region with a patterned contour effect using far fewer edges than full modeling of the actual terrain in three dimensions.

3.1.1 TONAL-VARIATION CONTOUR LIMITATIONS

The results and computational efficiency of the flat-surface, tonal-variation technique appear quite impressive initially. As some background prior to discussing the approach selected for this study, some of the limitations associated with it will be covered.

Figure 25(a) shows a model of a simple contour, 200 feet wide and 100 feet high. The illumination is from the upper left—so the left face is light and the right face is dark. Assume we are viewing this contour from an altitude of 350 feet, directly above its center. We will see it as shown in Figure 25(b). Figure 25(c) shows a figure defined entirely in the plane of the surface. Viewed from the same viewpoint, it will appear identical to the full three-dimensional representation of Figure 25(a).

Now shift the viewpoint 100 feet to the left, still looking straight down. Figure 25(d) shows the resulting image. If we wish a surface definition to give the correct image for this new viewpoint, it must be redefined to agree with

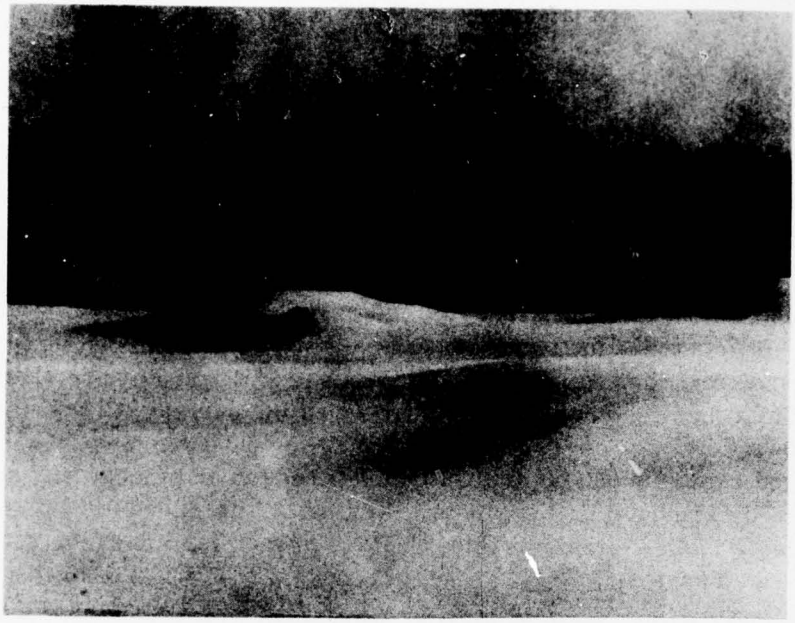


Figure 23. Contours by Terrain Modeling

Figure 24. Contours by Tonal Variation

Figure 25(d). If we do not redefine it, and if we interpret the resulting display based on knowledge that the feature is 100 feet high, then it will appear as though its three-dimensional definition is as shown in Figure 25(e).

In summary, a surface definition may correspond to a three-dimensional definition from a single viewpoint. If the surface definition remains unchanged as the viewer moves, there are two possible interpretations that will fit the sequence of images he sees. One is the flat tonal pattern defined—not a contour at all. The other is a contour that changes shape as he moves. Neither is the desired effect. The only situation in which reasonable satisfactory results might be expected to occur is if tonal patterns on a spatial surface are used to simulate gently rolling clouds—which lack any well defined spatial identity. Such tonal patterns with gentle variation can be applied to ground surface but are really producing surface detail, not contour cues.

For the type of contour cues we are seeking, it is necessary that the pattern of images correspond to a solid contour with true three-dimensional definition. This can be done with surface patterns only by redefining them with each change of viewpoint—for each new scene, in general. While this will still not allow creation of the effects of very low altitude flight where such contours will extend above the horizon, it may have merit if it can be shown to be computationally efficient and to produce effects with fewer edges than full contour definition.

3.1.2 APPROACH SUMMARY

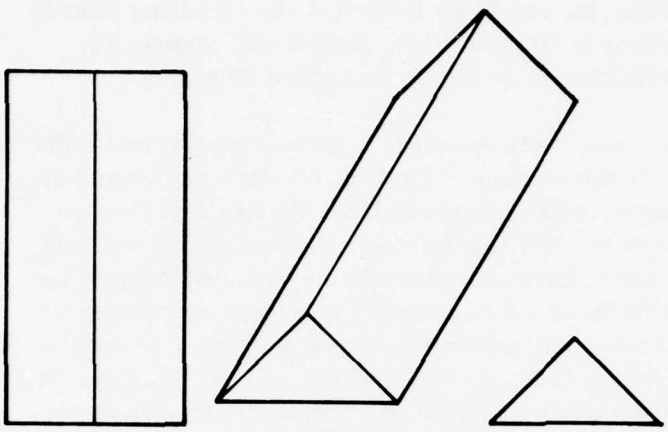
A test data base was prepared that included contours modeled in three dimensions. Scenes were made of this data base. A surface definition of the same contours, valid for a specific viewpoint, was prepared. Scene generation demonstrated the equivalence of the results. Analysis of the computational and memory burden of each approach failed to demonstrate any advantage for the surface definition. Subsequent effort was applied to evaluation of various approaches using full three-dimensional definition of contours.

3.2 ALGORITHMS DEFINED

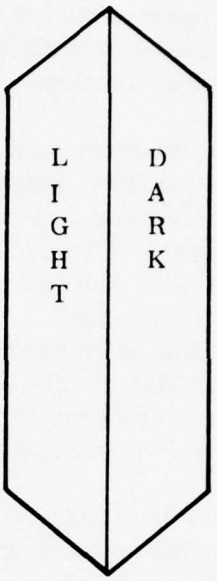
Full appreciation of the significance of some of the evaluation results requires precise understanding of the meaning of two algorithms, the Gradient Algorithm and the Curvature Algorithm, which have not always been distinguished in the past.

3.2.1 GRADIENT ALGORITHM

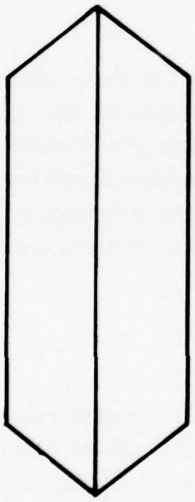
The term "Gradient Algorithm" will refer to the capability currently present in many hardware and software scene generation systems that allows tone or color to be incremented along an edge and to be incremented element by element as a scan line traverses a face. Thus, the tone of any face can be a linear



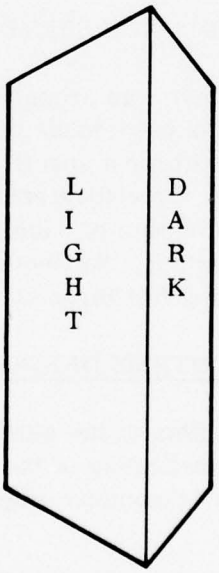
(a) 3-D DEFINITION



(b) IMAGE OF (a) FROM OVERHEAD

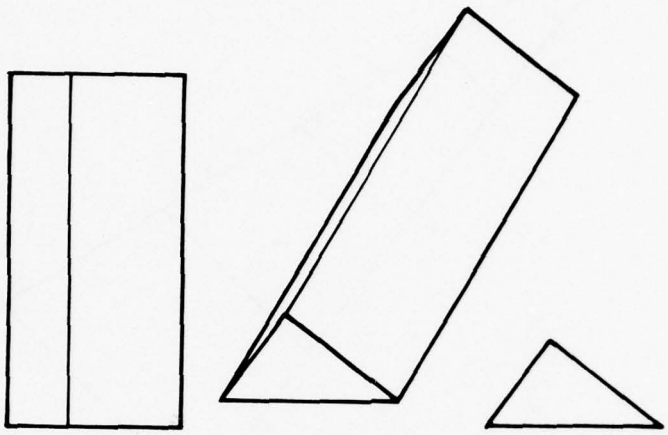


(c) SURFACE DEFINITION TO GIVE IMAGE (b)



(d) IMAGE OF (a) FROM LEFT OF OVERHEAD

Figure 25. Contour Representations and Images (Sheet 1 of 2)



(e) 3-D DEFINITION WHICH, FROM LEFT OF OVERHEAD, WOULD GIVE SAME IMAGE AS SURFACE DEFINITION (c)

Figure 25. Contour Representations and Images (Sheet 2 of 2)

function of location in the view window. This algorithm can be used to achieve a variety of effects. Figure 26, for example, shows a scene looking down on a carrier. Its wake is defined by a set of faces changing from white to the blue of the ocean. On systems without the gradient algorithm, the edges separating these faces are very visible and detract from realism. The gradient algorithm can be applied to achieve a gradual transition from white to blue, rendering the edges invisible. Note that no impression of curvature is intended or produced.

3.2.2 CURVATURE ALGORITHM

The "Curvature Algorithm" uses the gradient algorithm. If a set of conditions is met in modeling an object and in assigning tones to the object vertices, and if the gradient algorithm is then applied to the object, an excellent simulation of a curved object is produced. To produce this curvature requires not only the gradient algorithm but full application of the modeling and tonal assignment rules as well. These include such requirements as limitations on the angles between adjacent faces, limitation on the maximum image size of a single face on the display, tonal assignment or color brightness at a vertex based on angle between an illumination vector and the normal to the surface being approximated, etc. If these rules are violated, various interesting effects are produced, but valid simulation of curved objects cannot be expected.

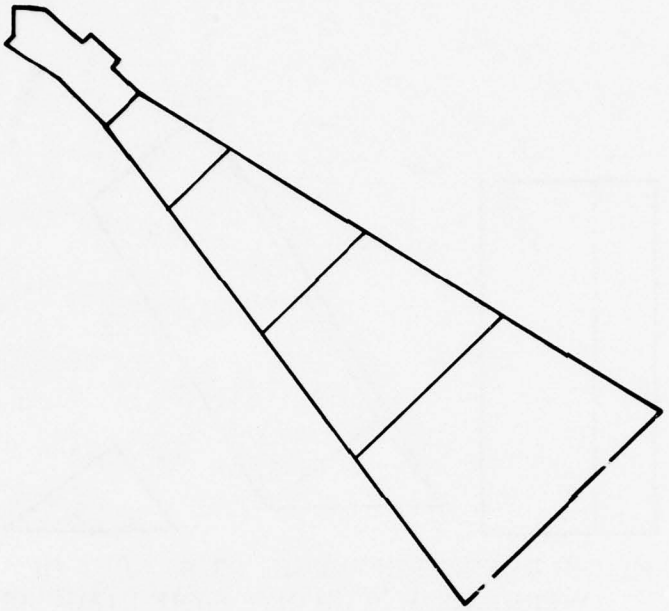


Figure 26. Carrier with Wake

3.3 PRELIMINARY RESULTS

A data base modeled for algorithm development use contains two discs and a sphere in addition to standard faces and a positive and a negative contour. These are seen in the perspective drawing shown in Figure 27. Figure 28 is a photograph of the scene from the same viewpoint as Figure 27. The contours are 200 feet wide, 100 feet high or deep, and completely symmetrical—all slopes are the same.

The next step was to prepare a data base containing surface-defined contours that would be valid from this viewpoint. This is shown on Figure 29. The eye tends to make a three-dimensional interpretation of the positive contour, however, the contour "edges" shown are all entirely on the surface. They might better be called tonal delineators, since they are not truly physical edges in the environment. The scene generated from this data base was indistinguishable from Figure 28. If any portion of a contour had extended above the horizon, it would not have been possible to produce a valid scene in this way.

The computational effort involved in this transformation for the positive contour is not severe. Rays from the viewpoint to each positive contour vertex must be projected to the ground and the point of intersection determined. These points constitute the vertices of the surface definition. This does represent added

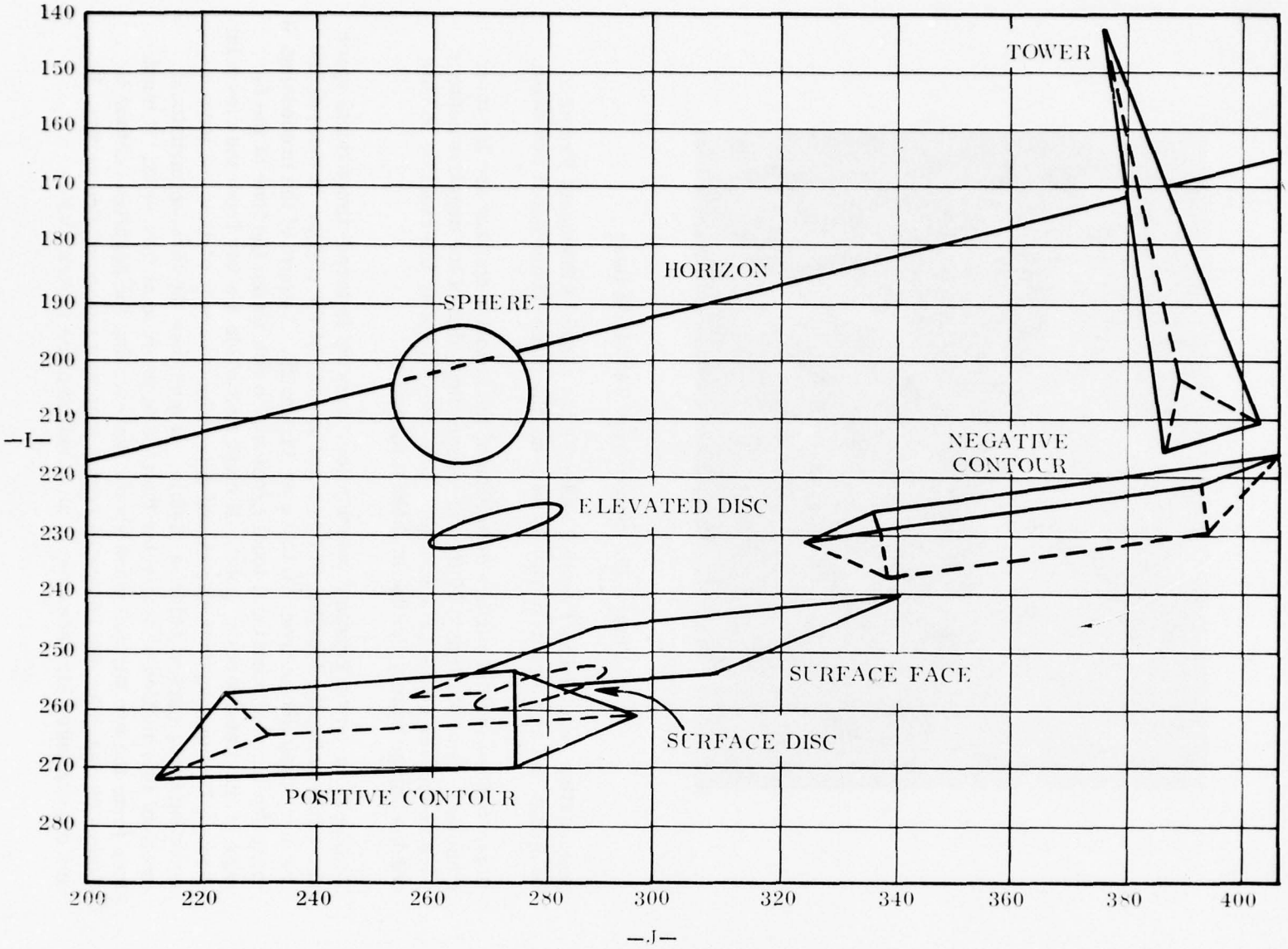


Figure 27. Texcon Test Scene from Viewpoint: 1000, 0, 500

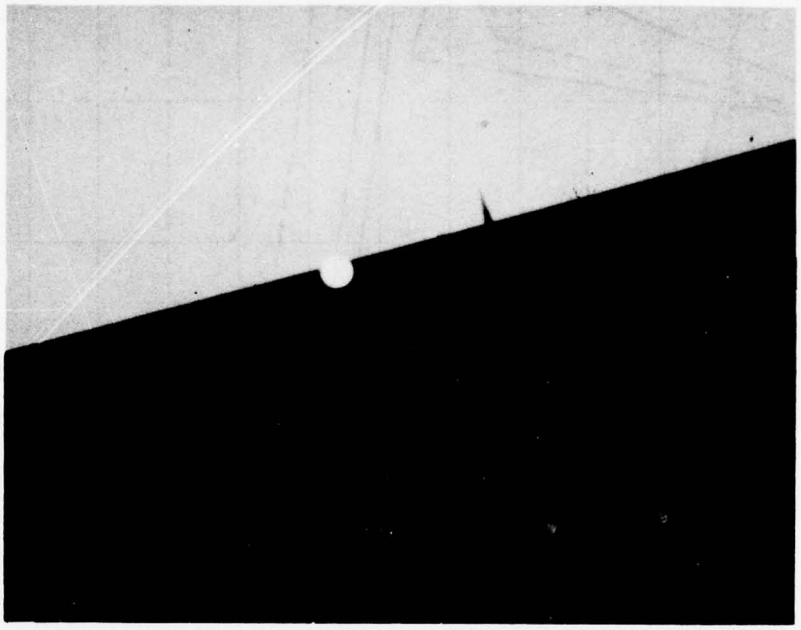


Figure 28. Scene from Test Data Base

computation required in Frame 2 and does not reduce subsequent Frame 2 computations as compared with processing the three-dimensional definition.

Computations for the surface definition of the negative contour are far more complex. Note on Figure 27 the far-descending edge of the negative contour goes behind the near-surface edge. This constitutes an intersection in view window image space, but not in three dimensions.

Frame 2, or a pre-Frame 2 computation, working in three-dimensional space must determine the location of this point on both of the edges. It must define the new edge, shown from Vb to Ve on Figure 29. As part of the processing, it must use three-dimensional tonal gradients to determine the tone of the far edge at the "intersection" point. It must next project a ray from the viewpoint through the near-surface vertex and determine where it strikes the long descending face, part of which is visible. It must use the three-dimensional gradient information to determine the tone of the face at this point. It must then form the two surface faces shown representing the negative contour in Figure 29 assigning the computed vertex tone to a new computed vertex, Ve, and the computed far-face tone to an actual surface vertex, Vd.

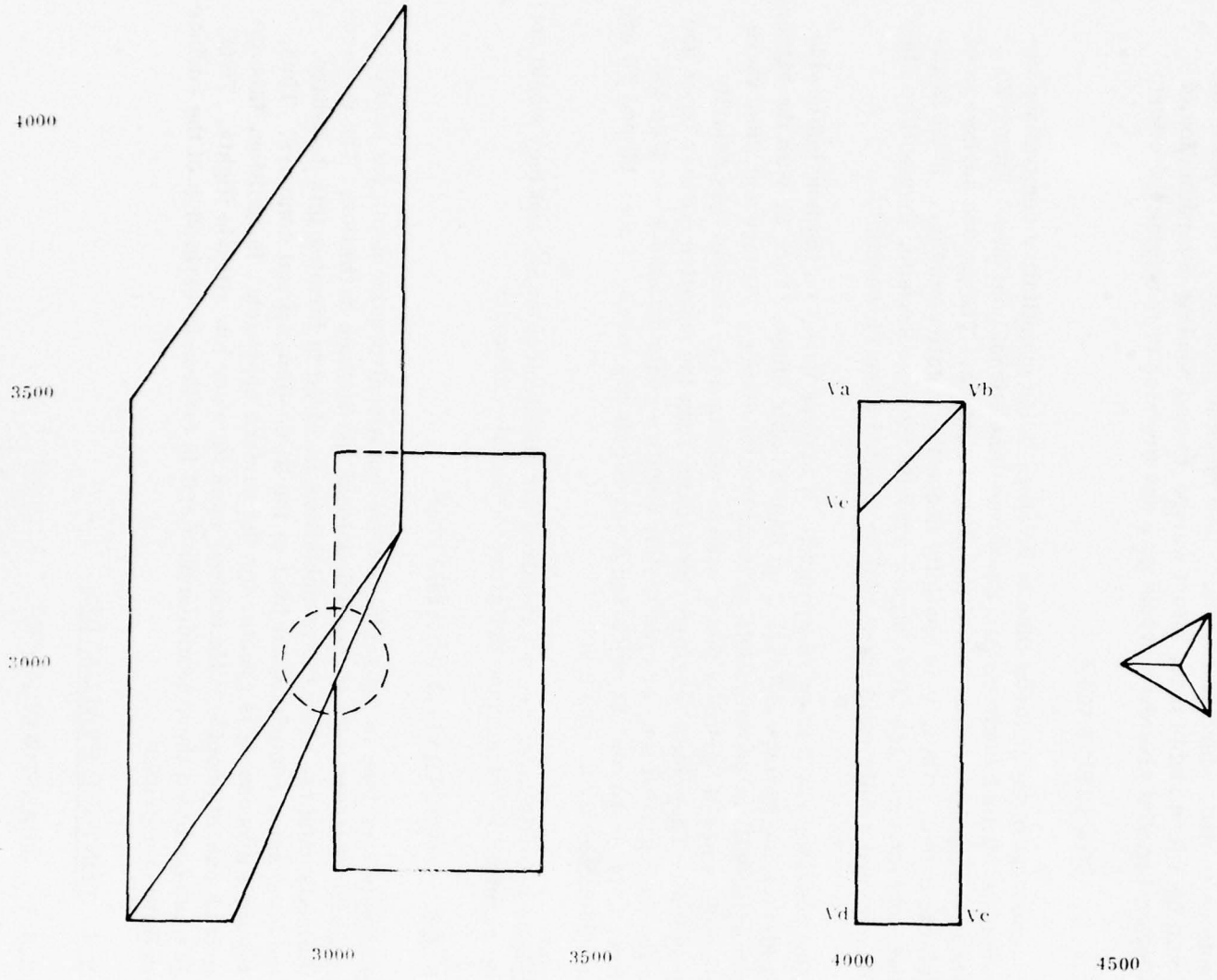


Figure 29. Surface Definition of Contours

The above was done manually in this exercise. An operational algorithm would have to validly handle in all cases the decisions as to what to do—which intersections to find, which tones, which new faces to generate, etc. Since this would be a formidable task, effort was postponed pending determination of whether benefits elsewhere would give any promise of justifying the effort.

3.3.1 PRIORITY FACES

The concept of the priority face is involved in the quantitative comparison between the surface contours and the three-dimensional contours. Figure 30 shows a negative contour below the ground surface. The ground surface is of infinite extent. Thus, it is spatially above the negative contour. If the negative contour faces are given higher priority than the surface, they will be displayed, and a fully valid scene will be created from viewpoint 1.

Now consider ray 2 from viewpoint 2. It strikes face 1 on the nonvisible side. It strikes the surface and face 2 on their visible sides. Face 2, with the higher priority, will be shown resulting in an invalid display. To prevent this, there must be a set of "priority faces" whose function is to correct this priority problem. They must have priority greater than the negative contour faces and must have ground tone or color. For negative contour faces 1 and 2 on the figure, faces 1A and 2A comprise a set of priority faces. Faces 1B and 2B are another choice for such a set.

These priority faces are not required for positive contours, and they would not be required in the surface definition of negative contours.

3.3.2 COMPARATIVE EVALUATION

Analysis was done to determine the computational requirements for positive and negative contours with three-dimensional and surface definition. The surface-defined contours, after the computations involved in forming this definition, have the same computational load as the three-dimensional contours. Thus, the total processing is greater for the surface approach. In addition, this approach cannot provide valid contour cues for very low altitude flights. Thus, it was concluded there would be no merit in further investigation of the surface-definition approach.

3.4 CONTOUR EVALUATION

3.4.1 TRANSITION FACES

Prior to describing the data bases from which evaluation scenes were made, the concept of transition faces must be covered.

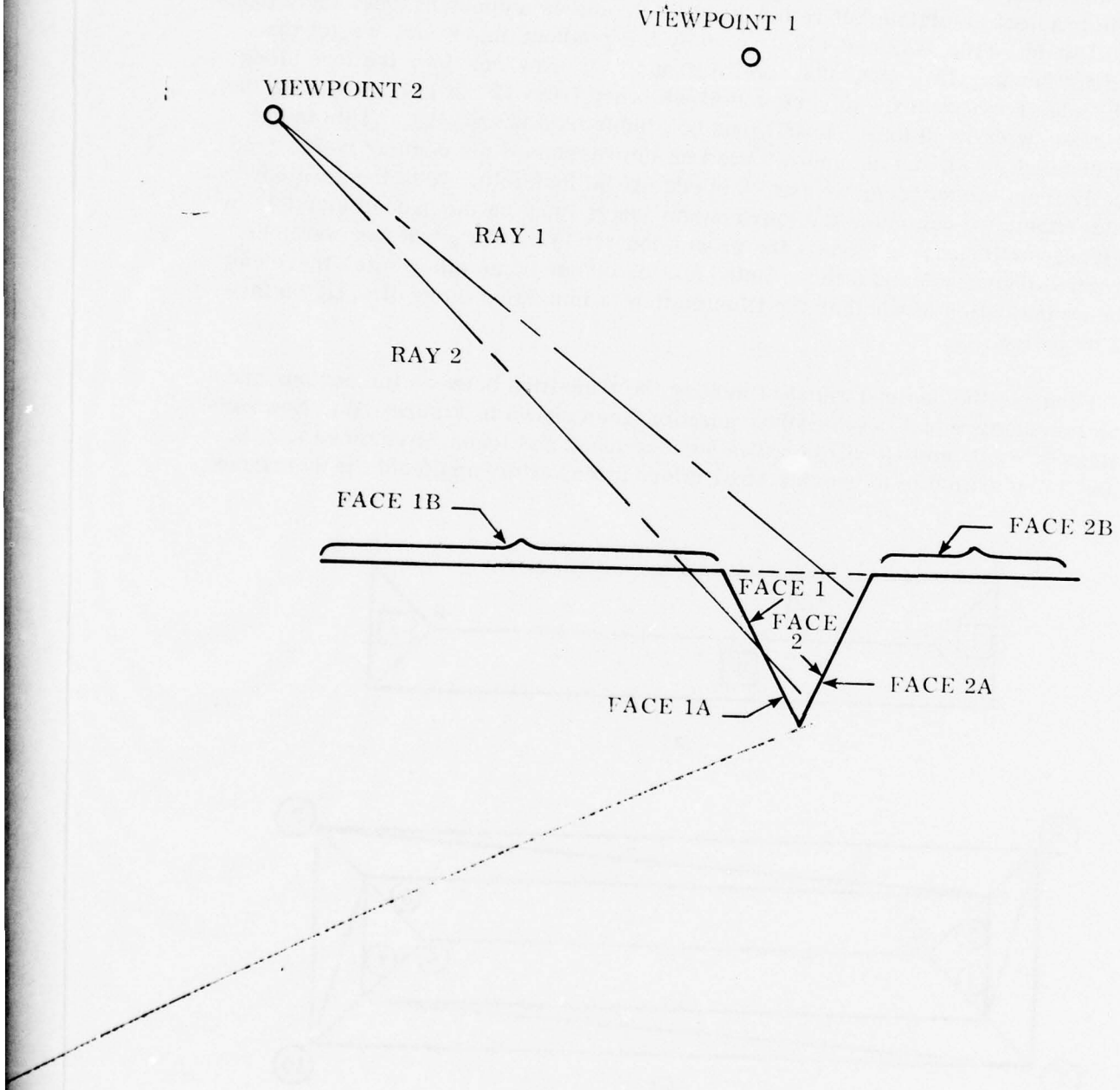
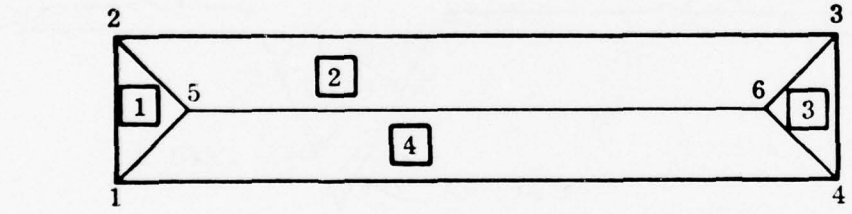


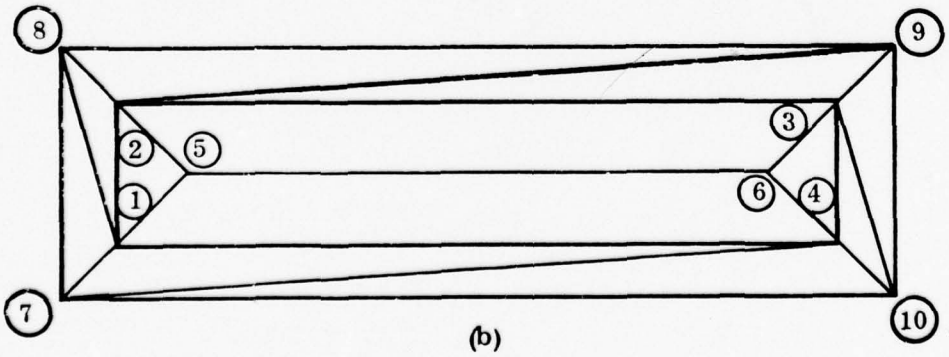
Figure 30. Priority Face Requirement

Figure 31(a) is a view looking down on a contour defined with nine edges. Assume it lies on a ground surface that has a day-bright tone of 150, a day-dark tone of 0 and that we have a late-morning, mid-winter, illumination vector direction of (-0.4364, 0.2182, -0.8729). X, to the right on the figure, is East, Y is North, and Z is "up." The resulting surface tone is 140. If we do not use the gradient algorithm but rather show each face as a constant tone, their tones will be 98, 110, 144, and 133. To apply the gradient algorithm, we get the vertex tones: 127, 119, 136, 144, 127 and 147. Now consider the tone along the edge from vertex 1 to vertex 2—it will vary from 127 to 119. It lies on the surface with a 140 tone. It will thus be visible as a sharp edge. This is not necessarily bad, but assume we want an impression of the contour rising gradually from the surface; we want this edge to be invisible. With the gradient algorithm, we can make the four contour edges lying on the ground invisible by giving vertices 1, 2, 3 and 4 the ground tone of 140. Now, whether we make vertex tones for 5 and 6 those listed above, or use some other rule, there can be no indication of whether the illumination is impinging more directly on face 2 or on face 4.

To achieve the desired effect of making the transition between the contour and the surface gradual, we need the transition faces shown in Figure 31(b). Now vertices 7, 8, 9, and 10 will be given surface tone, and tones for vertices 1, 2, 3, and 4 are available to provide tonal information validly applicable to their faces.



(a)



(b)

Figure 31. Contour With and Without Transition Faces

A question that might arise is whether we could form a single transition face with vertices 2, 8, 9, and 3; rather than the two shown. Consider the tones for these vertices: 119, 140, 140, 136. These values do not fit a linear function of position on the face; hence, they will not be shown properly by the gradient algorithm. Triangular faces cannot encounter this problem.

3.4.2 EVALUATION DATA BASES

3.4.2.1 Cont 1

Contours were modeled with flat faces (no gradient algorithm) with the contour having the same assigned tone as the ground with the illumination angle giving the perceived differences in tone. This tonal portrayal validly represents the spatial definition of the contours and provides a reference for comparison with other techniques. Figure 32 shows a view of Cont 1.

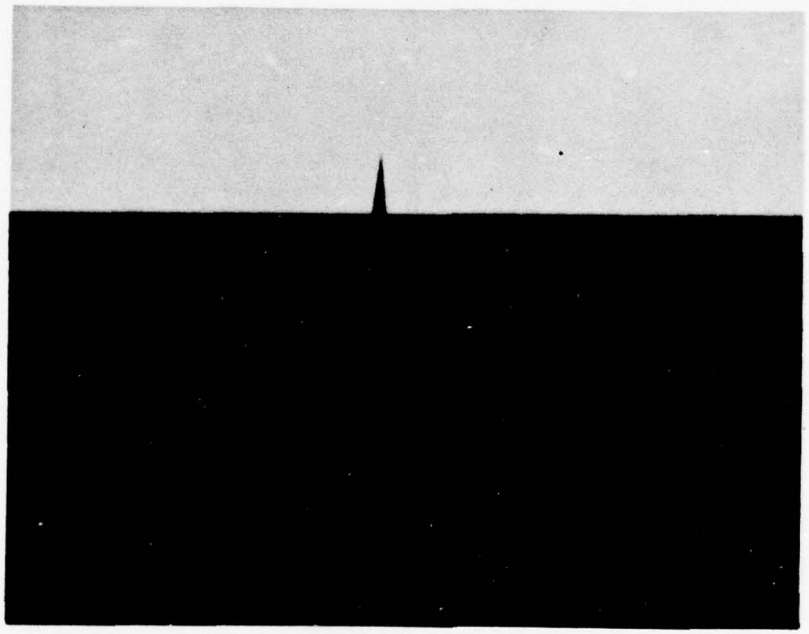


Figure 32. Flat-Face Contour Scene

3.4.2.2 Cont 2

The gradient algorithm was applied, but Cont 2 has no transition faces, similar to Figure 31(a). The vertex tones were assigned directly rather than being determined by the illumination vector. The capability for such assignment was built into the model to provide an additional experimental variable. The result is shown on Figure 33.

3.4.2.3 Cont 3

Transition faces were added to Cont 2. The negative contour needs priority faces as well. An attempt was made to improve efficiency by making the transition faces sufficiently large to serve as priority faces as well. This was part of the evaluation of the concept of using the gradient algorithm for purposes other than curvature simulation. As Figure 34 illustrates, the result is the appearance of tonal variation on the surface where no visible feature are intended.

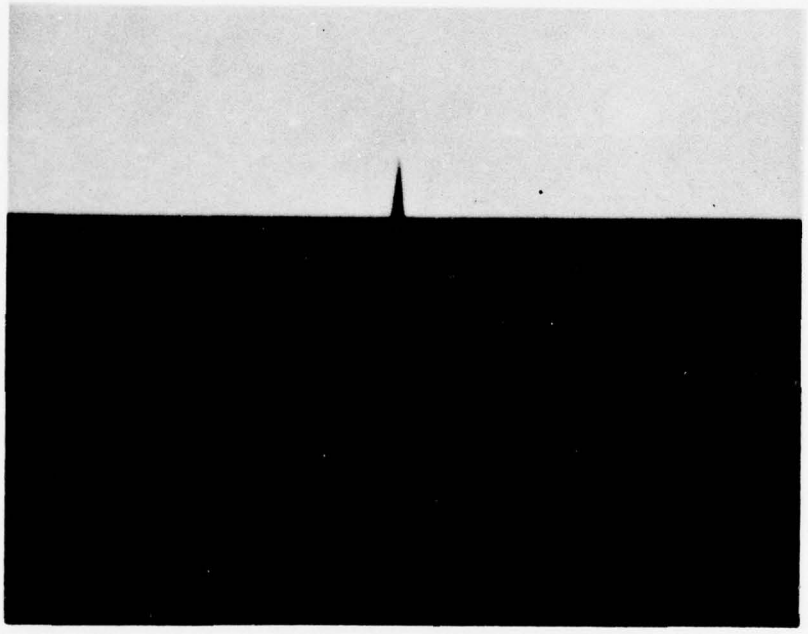


Figure 33. Contour Scene with Gradient and Assigned Tones

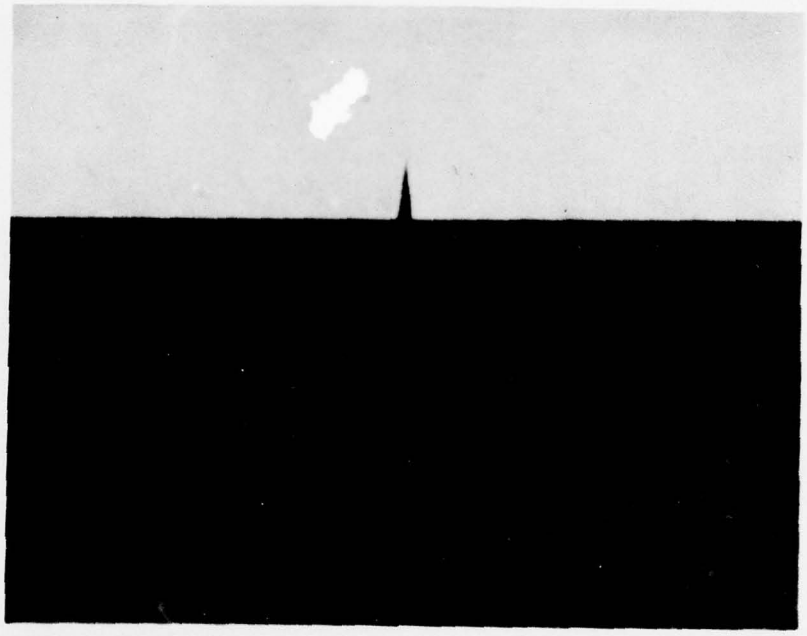


Figure 34. Contour With Large Transition Faces

3.4.2.4 Cont 4

The transition faces on the negative contour were reduced to form a band 100 feet wide around the contour, just as was earlier done for the positive contour. Tonal assignment was still arbitrary. As shown on Figure 35, results were much improved over Cont 3.

3.4.2.5 Cont 5

This has the same spatial definition as Cont 4, but tonal assignment as based on illumination direction and vertex normals assigned as in the curvature algorithm, as the average of the normals of the faces sharing the vertex. This scene gives rather impressive results, as seen on Figure 36. However, it also illustrates another of the things that can happen when the gradient algorithm is extended beyond its original purpose. The illumination is coming from the left, making the small left face quite bright and the small face on the right rather dark. Thus, the left vertices of the long face are bright and the right vertices dark. We thus have a tonal variation along the length of this face that has no relation to any intended spatial definition of the contour.

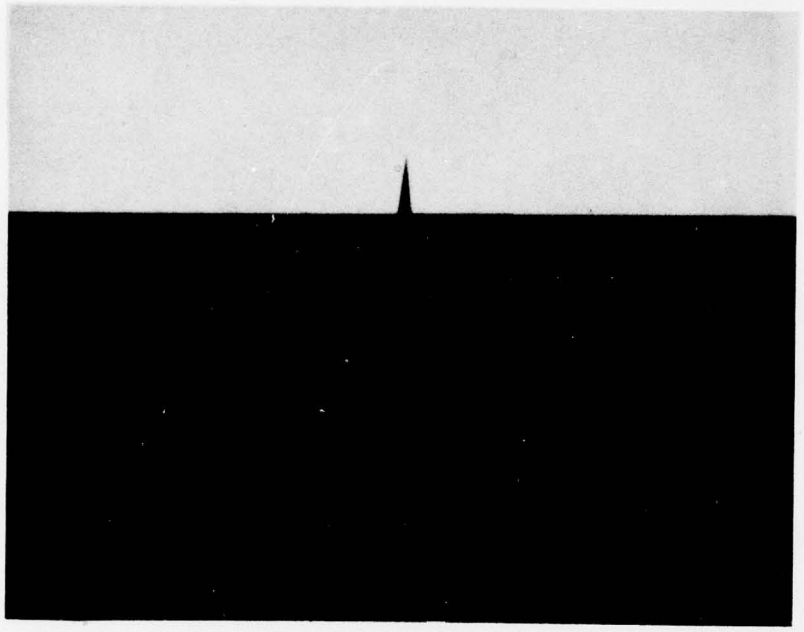


Figure 35. Small Transition Faces, Assigned Tones

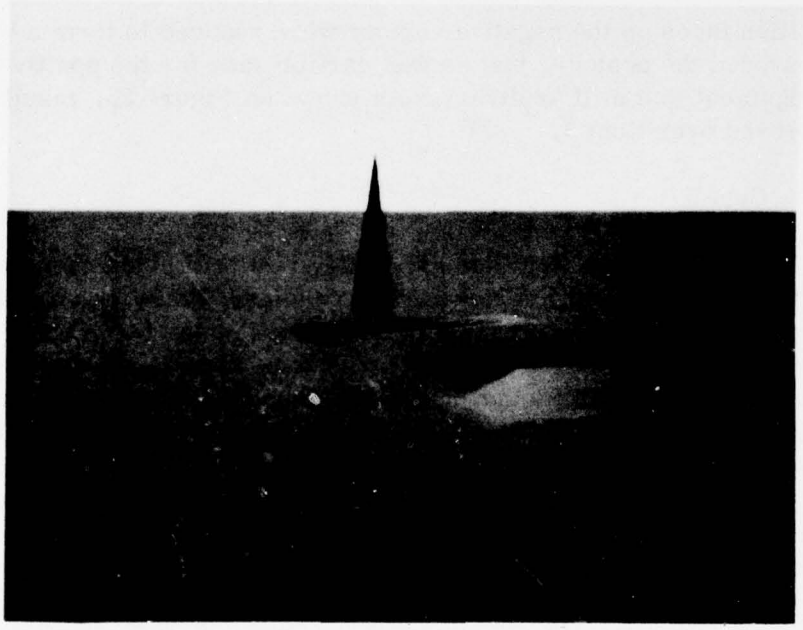


Figure 36. Small Transition Faces, Computed Tones

3.4.2.6 Cont 6

Cont 6 is the same as Cont 5 but without the transition faces. Comparing Figure 37 with Figure 36, we see the effect of this change—the sharpness of the transition between the surface and the contour.

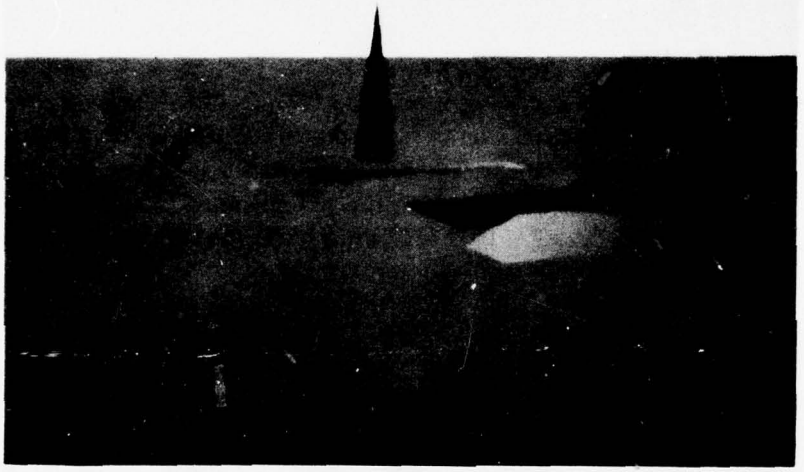


Figure 37. No Transition Faces, Computed Tones

3.4.2.7 Data Base Selection

Data bases Cont 1, Cont 5, and Cont 6 were selected for further evaluation based on the scenes discussed above. Cont 1 and Cont 6 are both minimum-edge definitions—nine edges each. Cont 5 with its transition faces has 21 edges. If large numbers of contours are used, this difference can be significant. Thus, if a gradient scheme is to be used, we should base the evaluation of the improvement due to these edges—the results of Cont 5 as compared with Cont 6—on more than a single pair of scenes.

It was therefore decided to produce scenes of these three data bases from a sequence of viewpoints—particularly including some giving a better view down into the negative contour. It was also decided to select the most meaningful viewpoint found and produce scenes with variations in the illumination vector.

Specific plans for modeling and producing scenes of connected strings of contours would be based on the evaluation of these results along with plans for video tape generation.

3.4.3 CONT 1 SEQUENCE

Figure 38(a) through 38(h) shows data base Cont 1 from a sequence of viewpoints. The contour is unrealistic—it is not curved—but it does give clear, unambiguous spatial information of its definition, its orientation, and the illumination direction. This sequence provides some far better views of the negative contour than were produced in earlier effort. Figure 38(b) was selected as the most meaningful location for evaluating the effect of varying illumination direction.

3.4.4 CONT 1 ILLUMINATION VARIATION

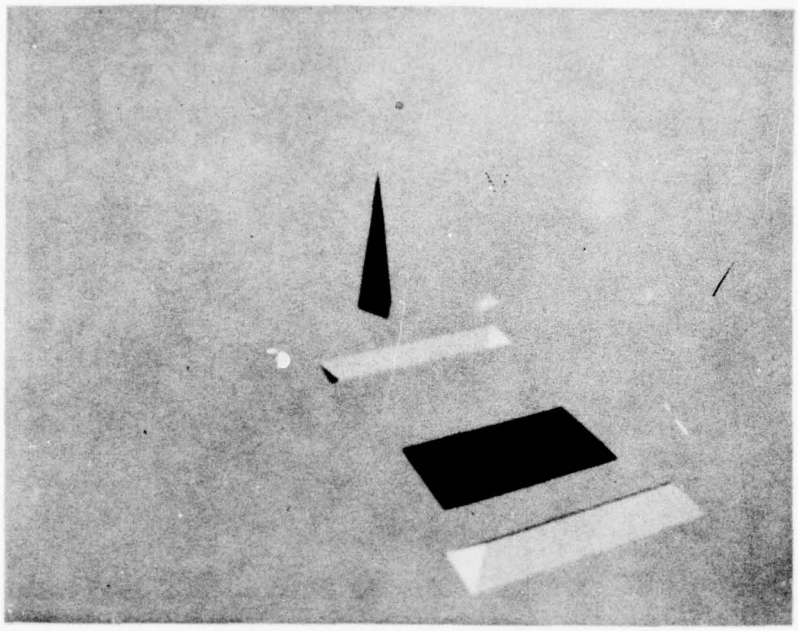
Figure 39(a) through 39(i) shows Cont 1 with the direction of the illumination vector varied in steps. The direct, unambiguous interpretation of the scene mentioned earlier is again apparent, with the exception of Figure 39(e) where the illumination direction is such that all faces have the same tone.

3.4.5 CONT 5 SEQUENCE

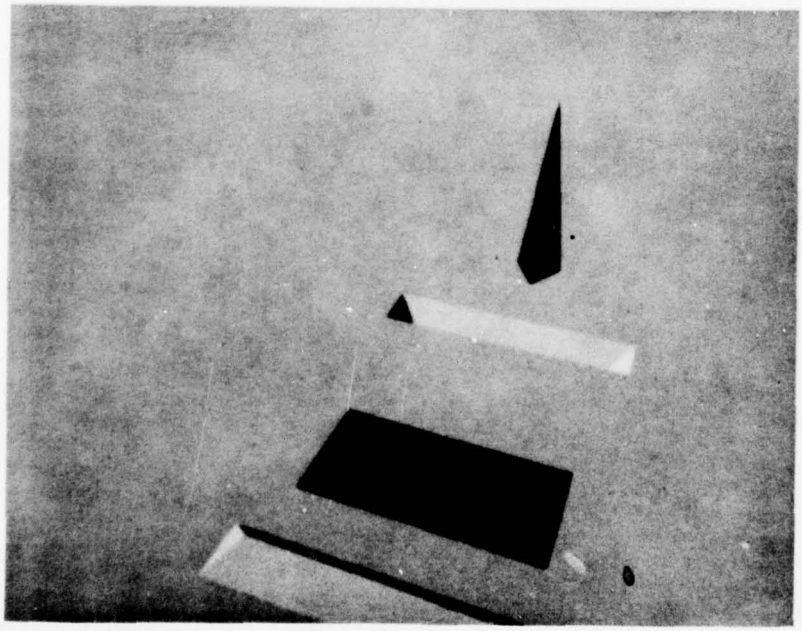
Cont 5 represents the opposite extreme from Cont 1—the added edges for transition are present, and the added processing for gradient implementation is applied. Figure 40(a) through 40(h) is a sequence taken from the same viewpoints as those used for the Cont 1 evaluation sequence. The smooth gradual tonal variation has appeal. However, just as when the full curvature algorithm is applied, the gradient processing does nothing to smooth the outline of an object against the background. The requirements of the full curvature algorithm generally result in an outline that is a reasonable approximation of the curved object being modeled. The edge constraints of this contour representation result in greater contrast between the curved-appearing interior and the obviously noncurved outline.

3.4.6 CONT 5 ILLUMINATION VARIATION

Figure 41(a) through 41(i) shows the same sequence of illumination variation as was used for Cont 1. The nature of the tonal variation along the long face, which cannot be correlated with any causative spatial variation, is noted to some degree for all illumination directions except for Figure 41(e), the "washout" scene.

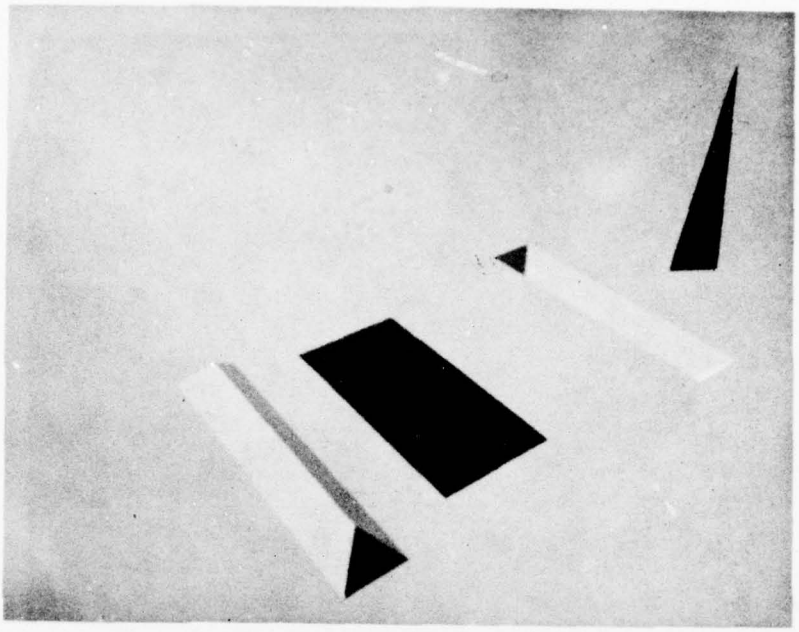


(a)

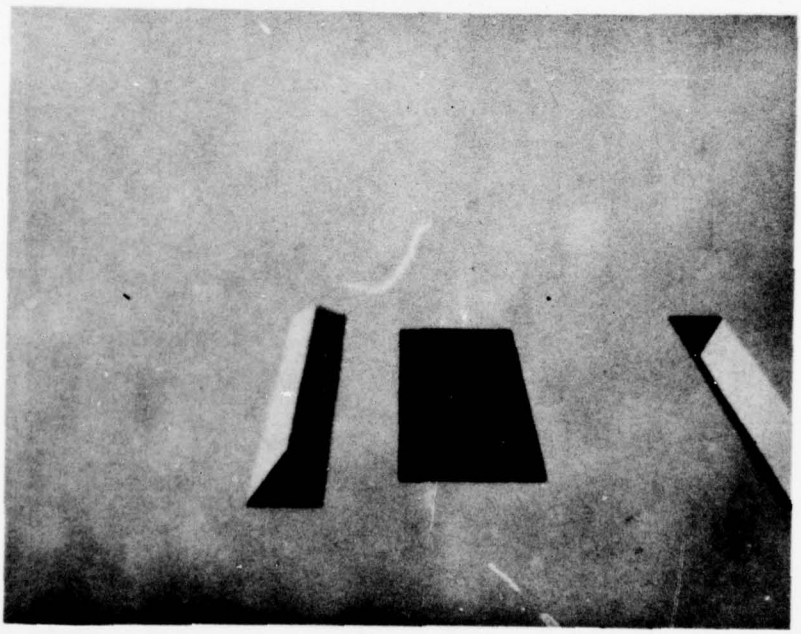


(b)

Figure 38. Cont 1 Data Base from Sequence of Viewpoints (Sheet 1 of 4)

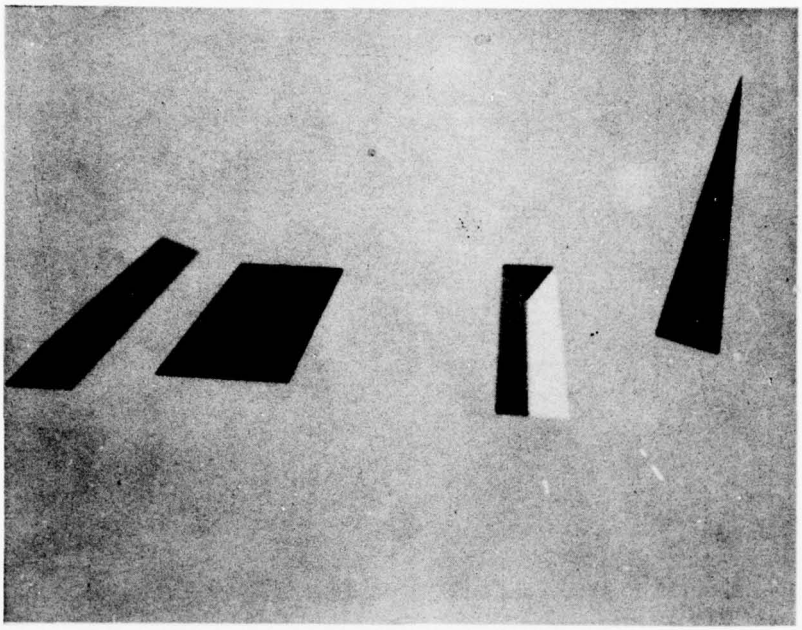


(c)

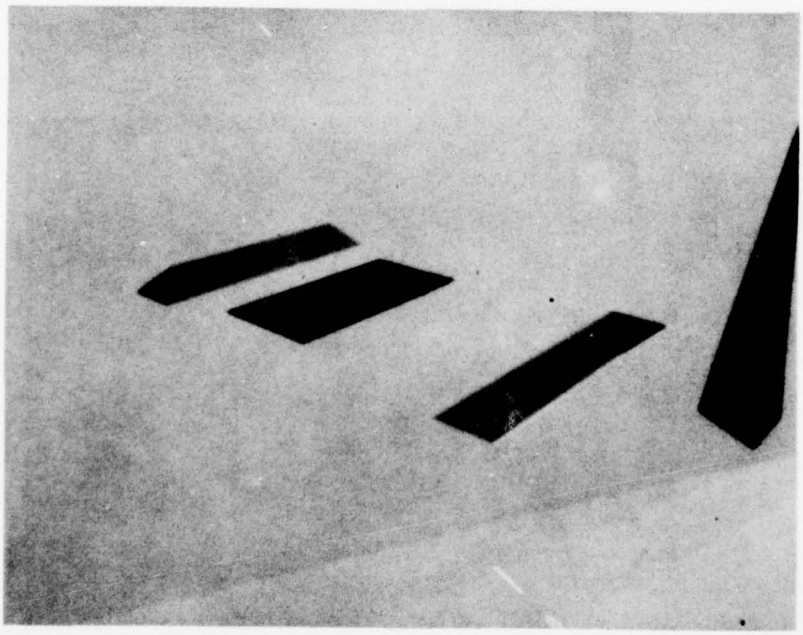


(d)

Figure 38. Cont 1 Data Base from Sequence of Viewpoints (Sheet 2 of 4)

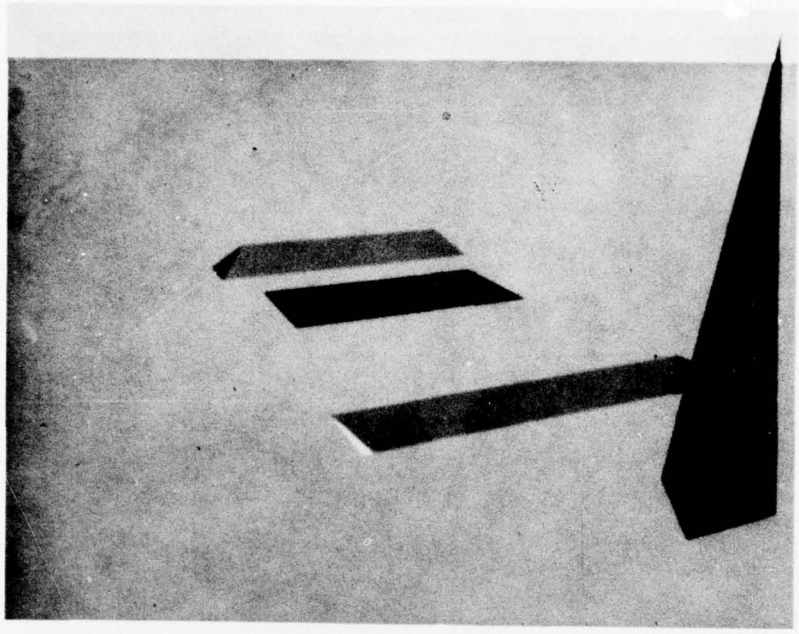


(e)

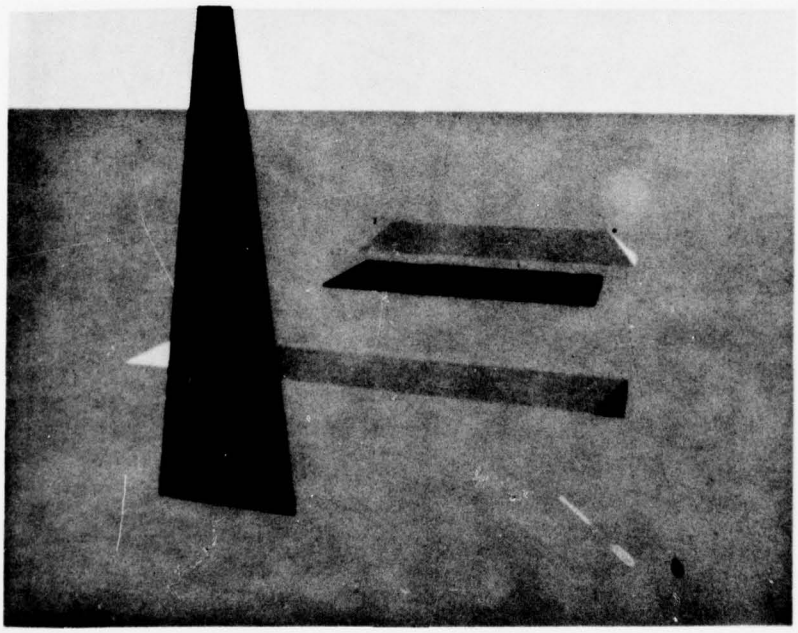


(f)

Figure 38. Cont 1 Data Base from Sequence of Viewpoints (Sheet 3 of 4)

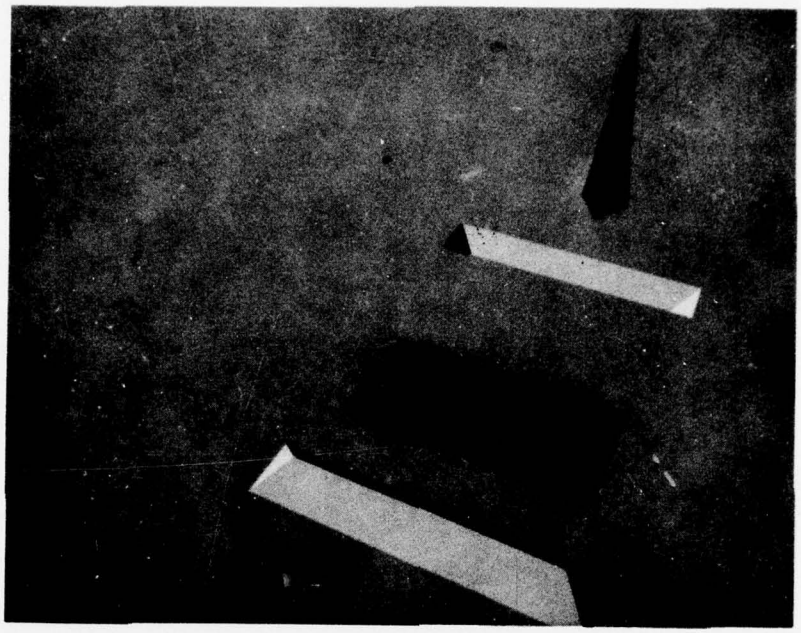


(g)

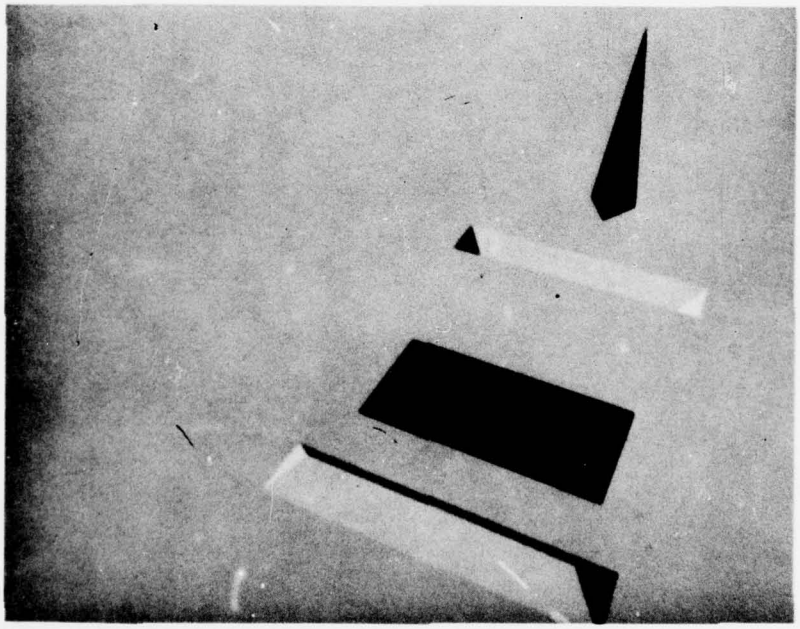


(h)

Figure 38. Cont 1 Data Base from Sequence of Viewpoints (Sheet 4 of 4)

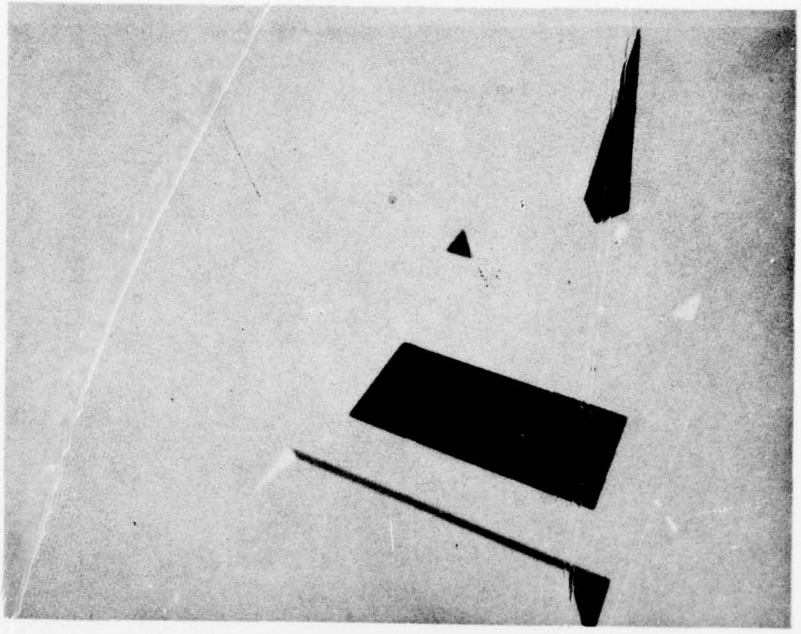


(a)

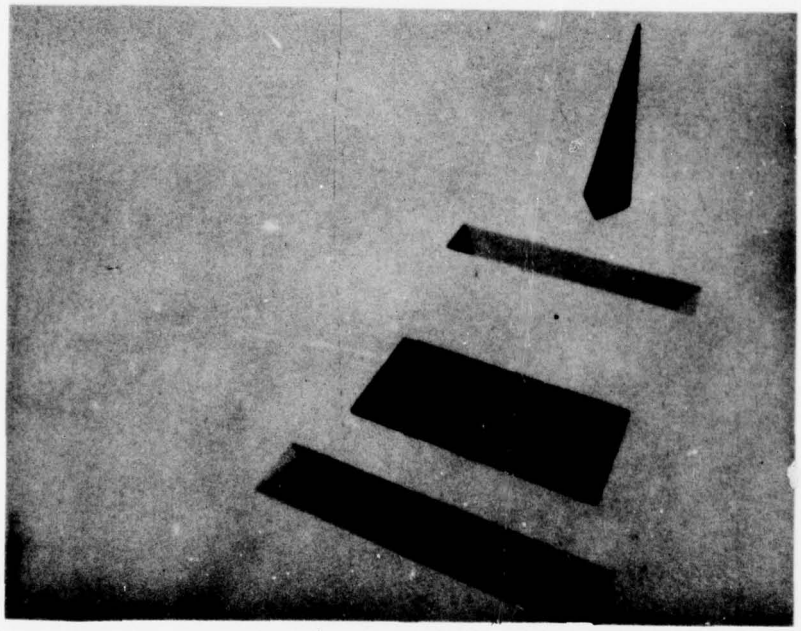


(b)

Figure 39. Cont 1 Data Base with Varying Illumination Direction (Sheet 1 of 5)



(c)



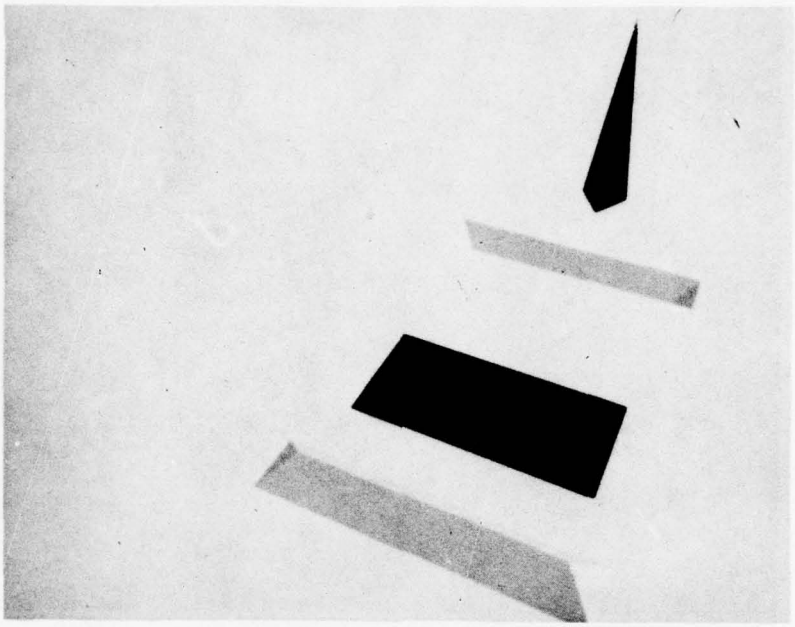
(d)

Figure 39. Cont 1 Data Base with Varying Illumination Direction (Sheet 2 of 5)

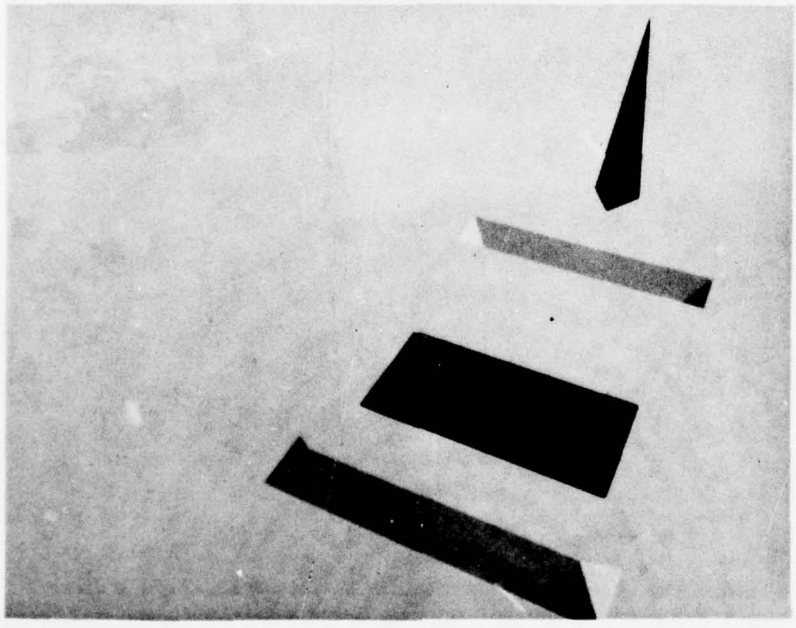
(e)

(f)

Figure 39. Cont 1 Data Base with Varying Illumination Direction (Sheet 3 of 5)

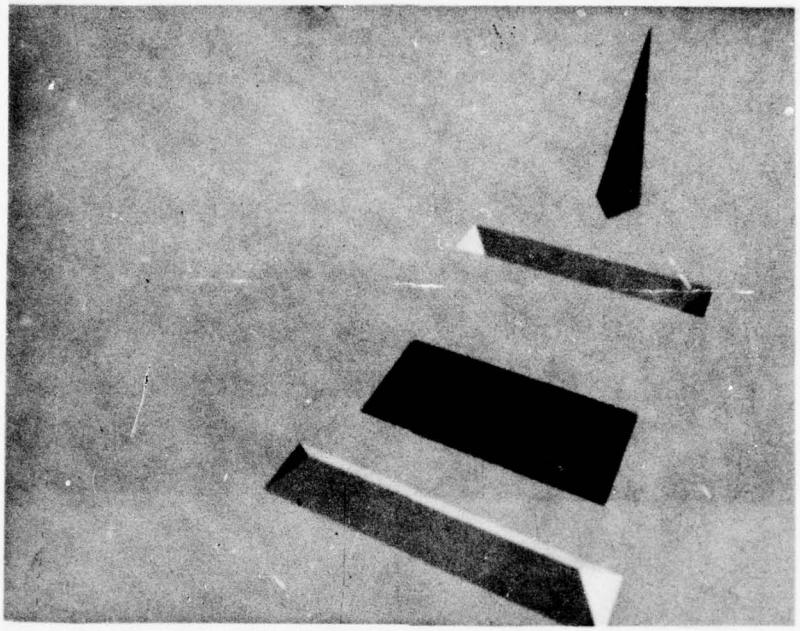


(g)



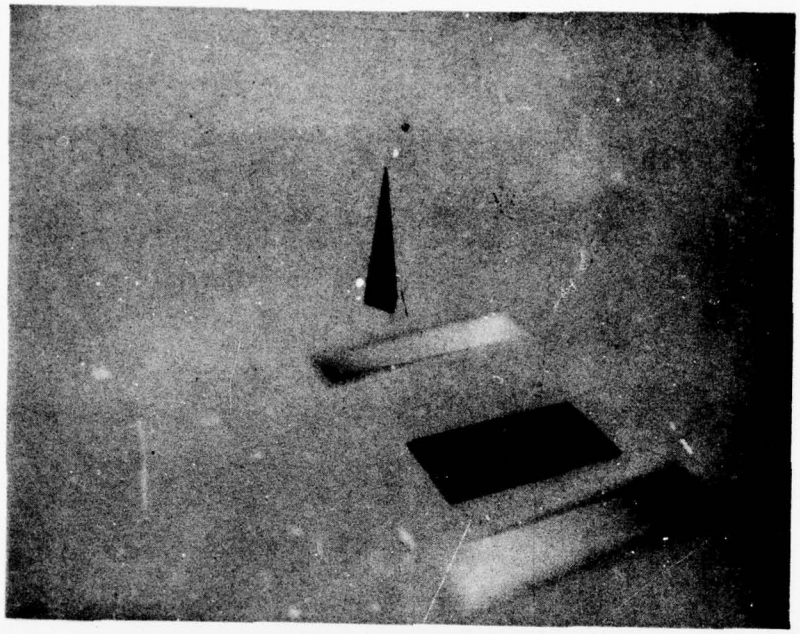
(h)

Figure 39. Cont 1 Data Base with Varying Illumination Direction (Sheet 4 of 5)

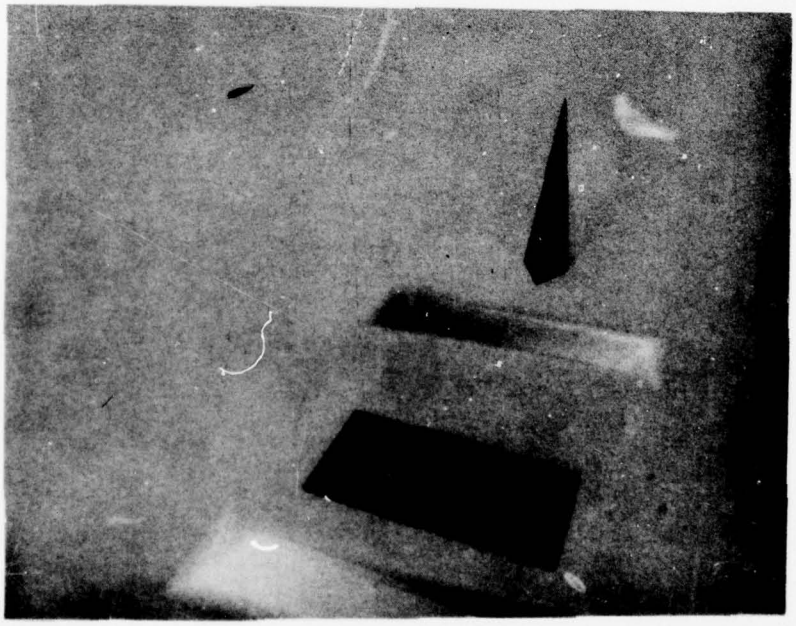


(i)

Figure 39. Cont 1 Data Base with Varying Illumination Direction (Sheet 5 of 5)

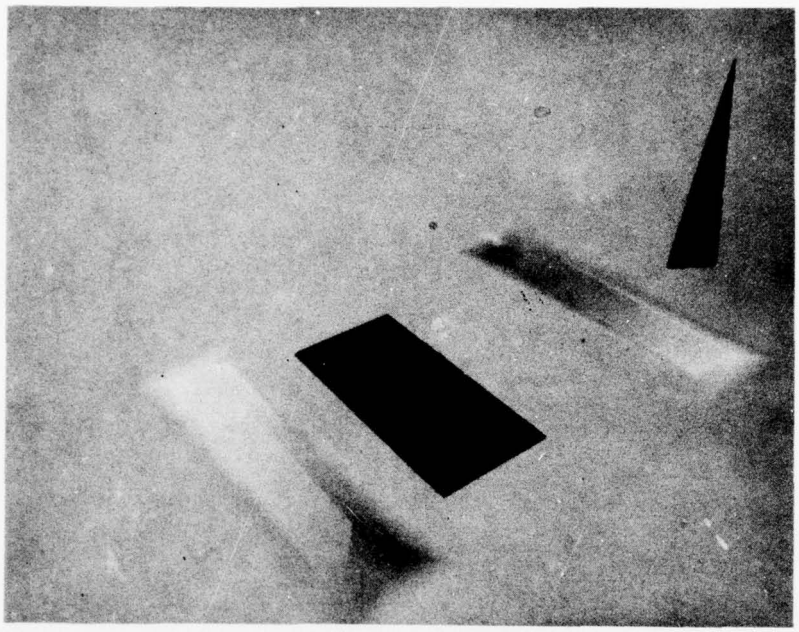


(a)

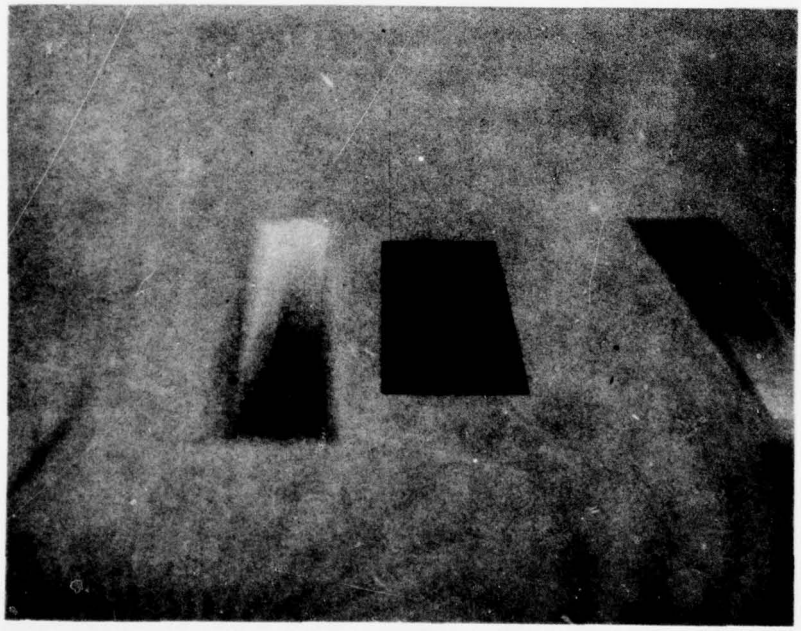


(b)

Figure 40. Cont 5 Data Base from Sequence of Viewpoints (Sheet 1 of 4)

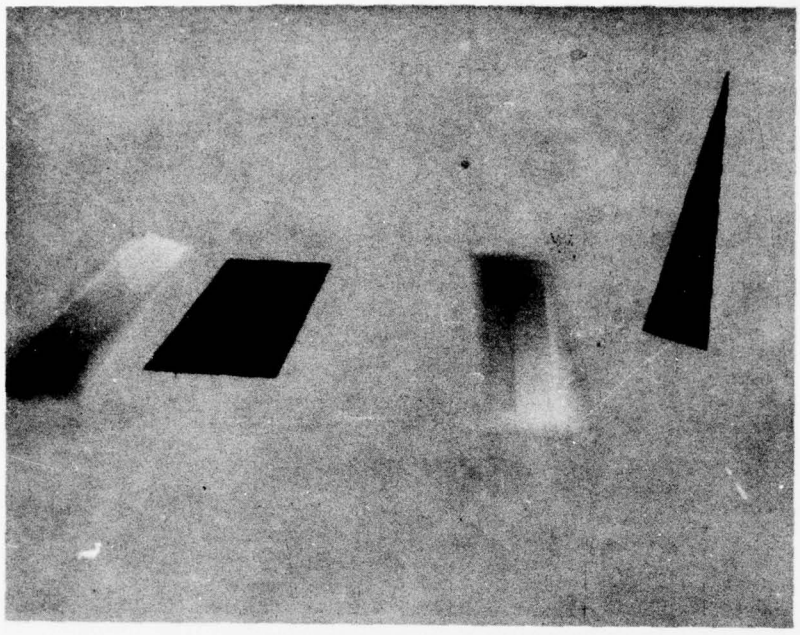


(c)

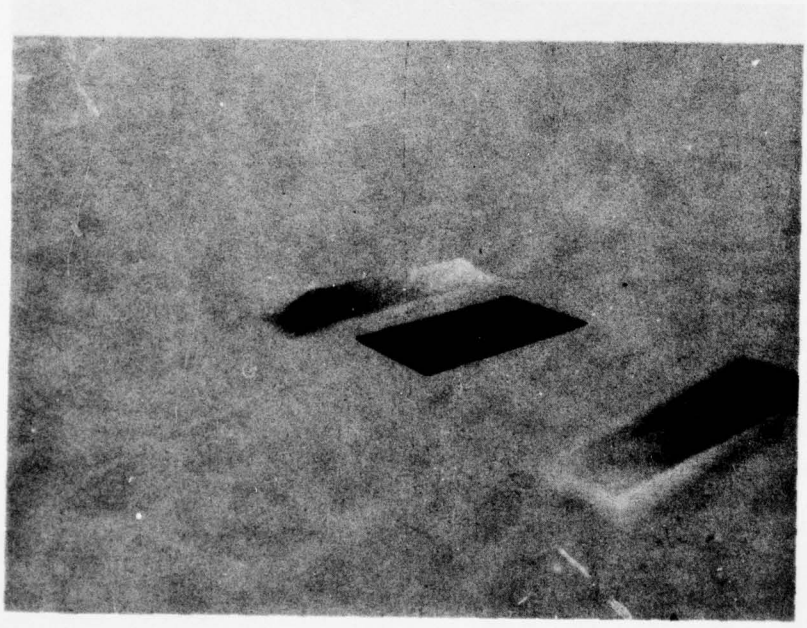


(d)

Figure 40. Cont 5 Data Base from Sequence of Viewpoints (Sheet 2 of 4)



(e)

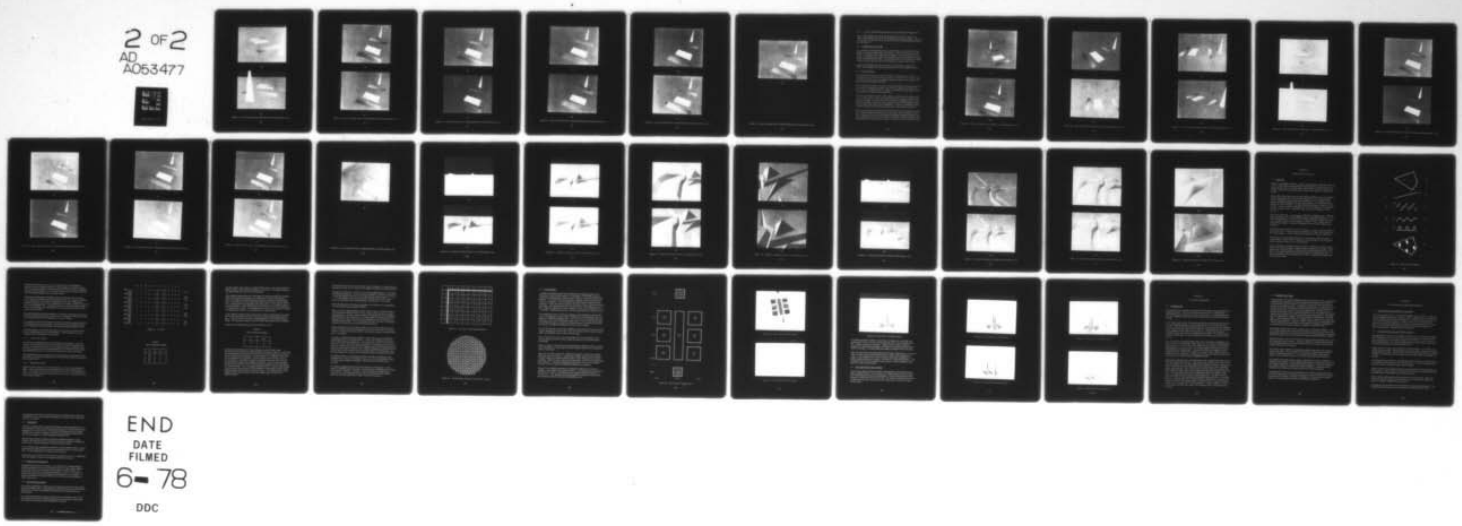


(f)

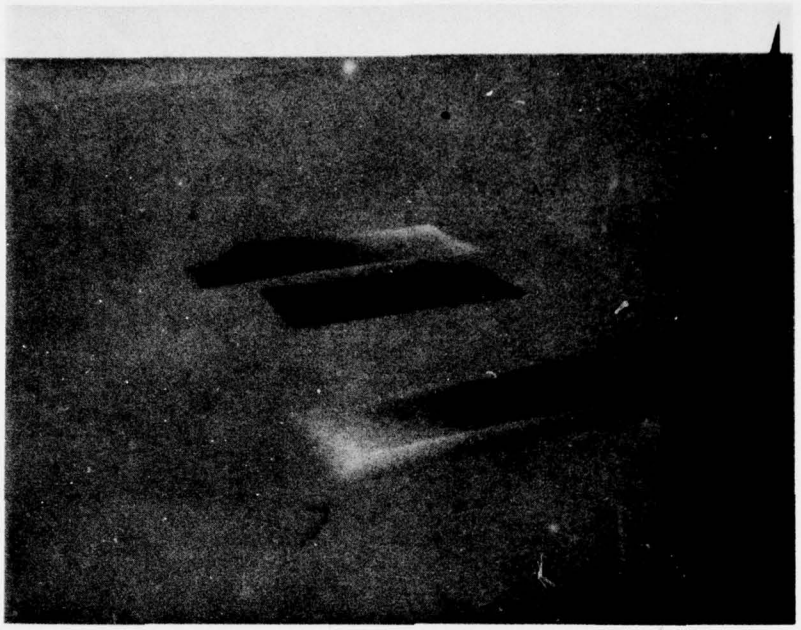
Figure 40. Cont 5 Data Base from Sequence of Viewpoints (Sheet 3 of 4)

AD-A053 477 GENERAL ELECTRIC CO DAYTONA BEACH FLA F/G 5/9
COMPUTER IMAGE GENERATION IMAGERY IMPROVEMENT: CIRCLES, CONTOUR--ETC(U)
SEP 77 W M BUNKER, N E FERRIS F33615-76-C-0038
UNCLASSIFIED AFHRL-TR-77-66 NL

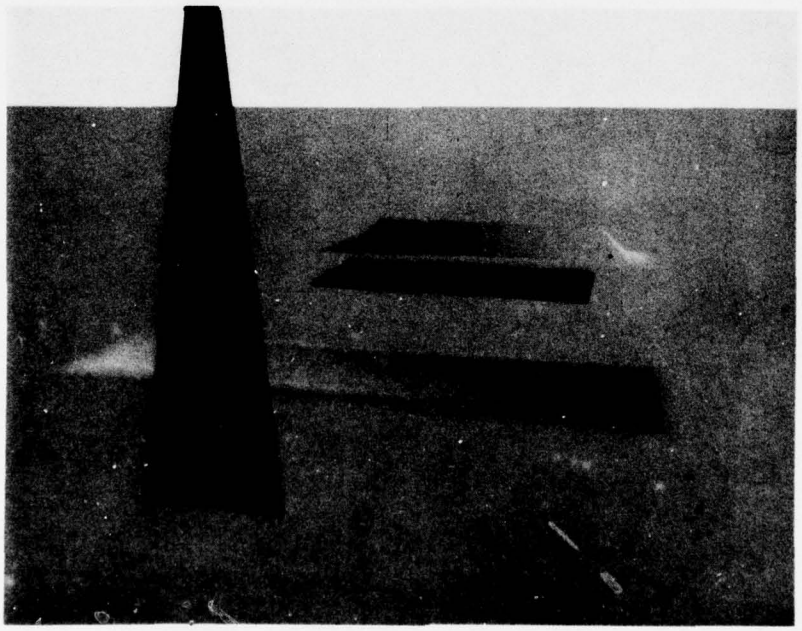
2 OF 2
AD
AD53477



END
DATE
FILMED
6-78
DDC

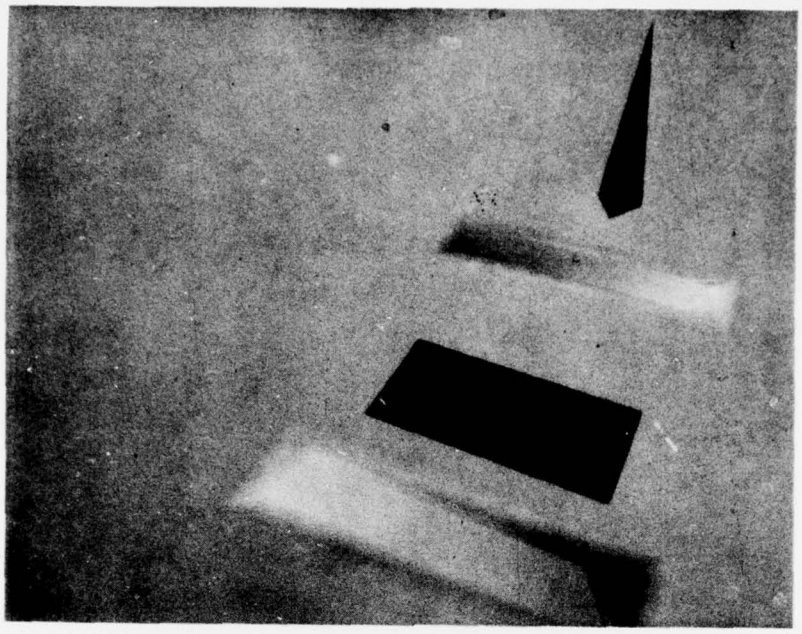


(g)

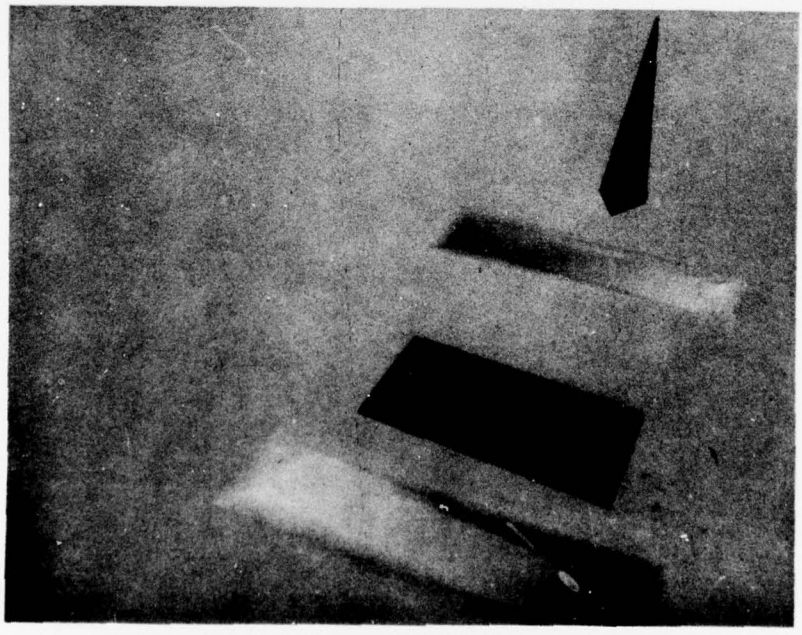


(h)

Figure 40. Cont 5 Data Base from Sequence of Viewpoints (Sheet 4 of 4)

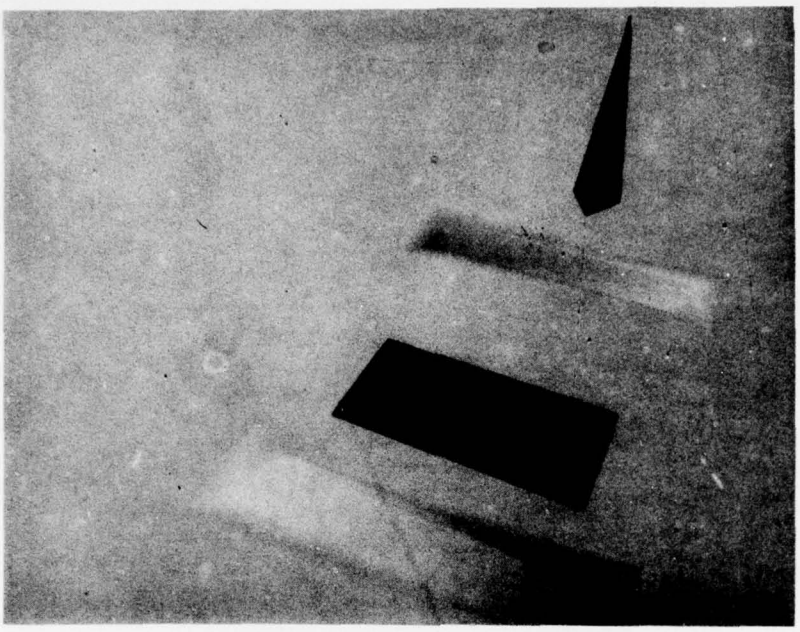


(a)

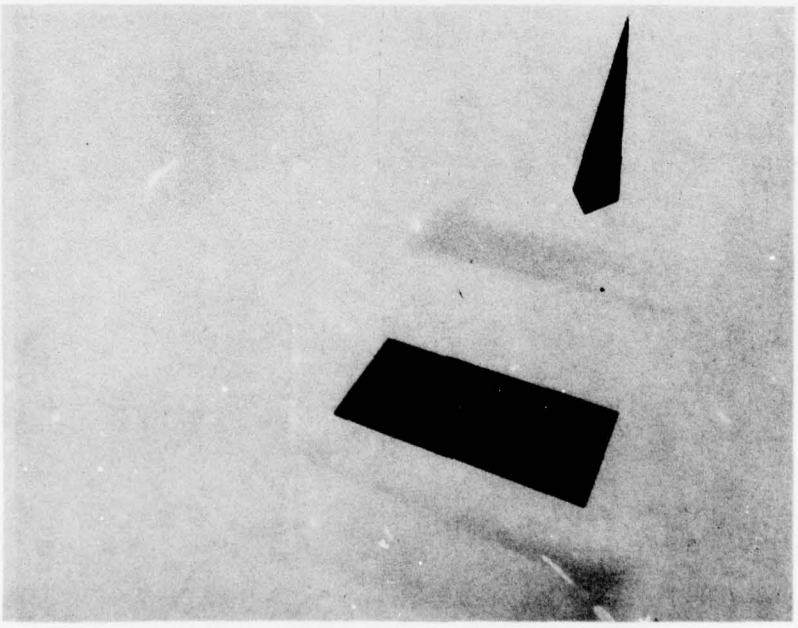


(b)

Figure 41. Cont 5 Data Base with Varying Illumination Direction (Sheet 1 of 5)

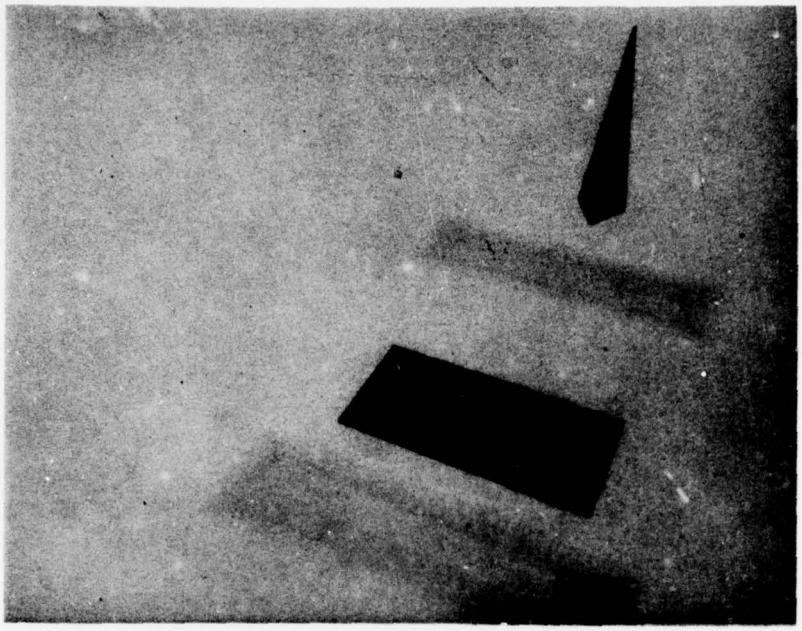


(c)

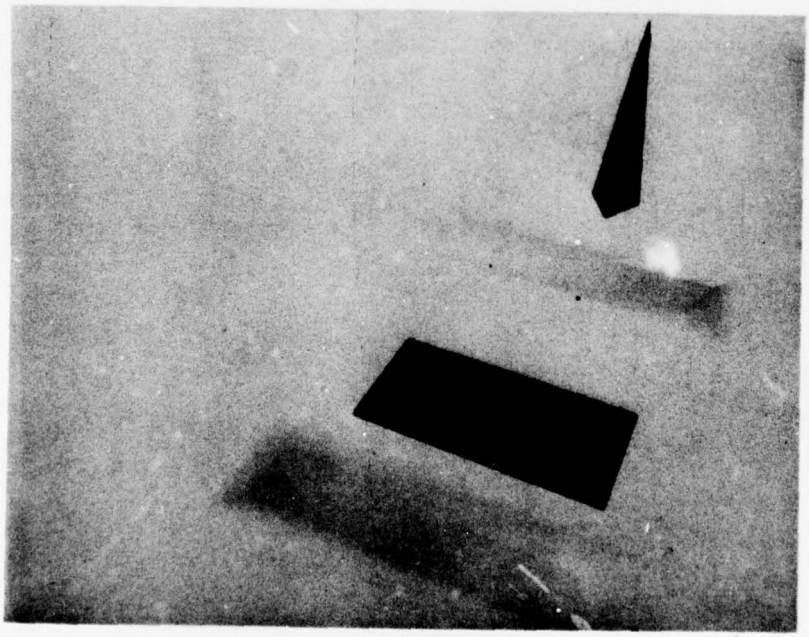


(d)

Figure 41. Cont 5 Data Base with Varying Illumination Direction (Sheet 2 of 5)

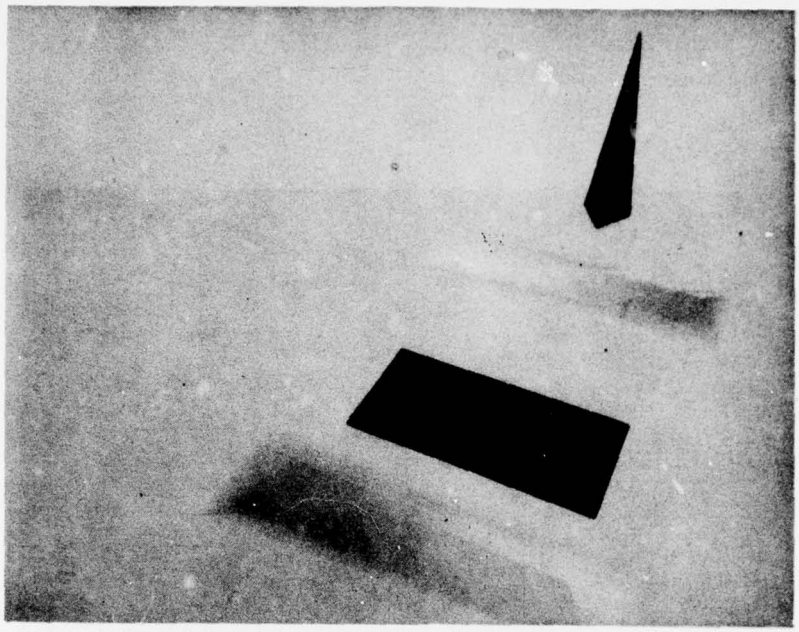


(e)

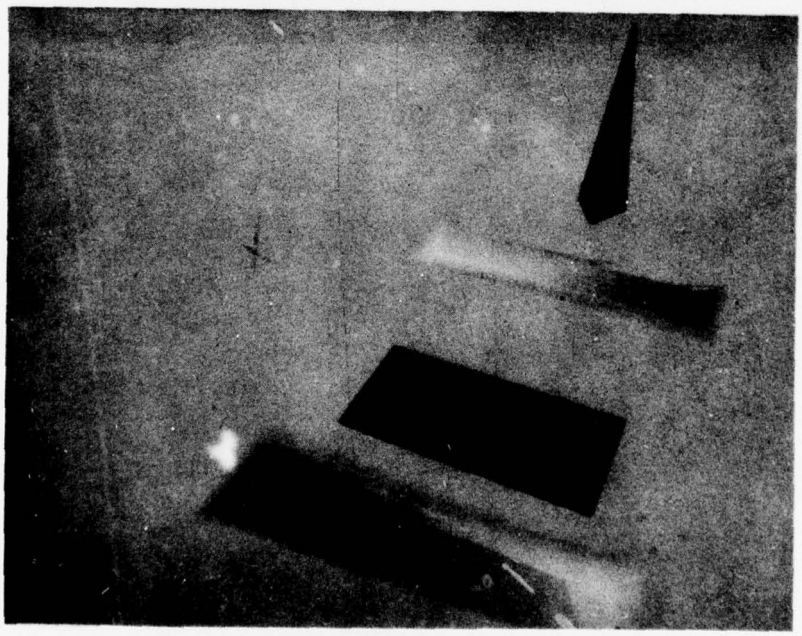


(f)

Figure 41. Cont 5 Data Base with Varying Illumination Direction (Sheet 3 of 5)

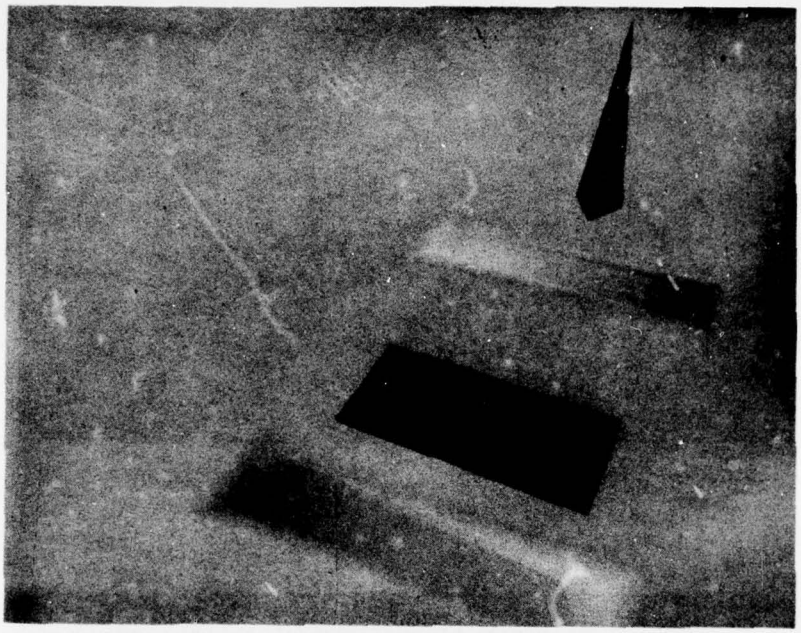


(g)



(h)

Figure 41. Cont 5 Data Base with Varying Illumination Direction (Sheet 4 of 5)



(i)

Figure 41. Cont 5 Data Base with Varying Illumination Direction (Sheet 5 of 5)

3.4.7 CONT 6 VIEWPOINT SEQUENCE AND ILLUMINATION SEQUENCE

Figure 42(a) through 42(h) shows the viewpoint sequence for Cont 6, and Figure 43(a) through 43(i) shows the illumination direction sequence. They are, as expected, the same as Cont 5 except for the more abrupt transition with the ground surface.

3.5 CONNECTED CONTOURS

If a reasonable representation can be generated they can obviously be strung together to create winding ridges and valleys. Data bases were prepared to produce evaluation scenes of this use of contours. Two data bases were prepared, one with flat-face contours and the other with vertex normals for application of the gradient algorithm. No transition faces were used in the gradient version—earlier comparisons show the difference made by these faces.

Figure 44(a) through 44(h) shows several views of the flat version, and Figure 45(a) through 45(h) shows comparable views of the gradient version.

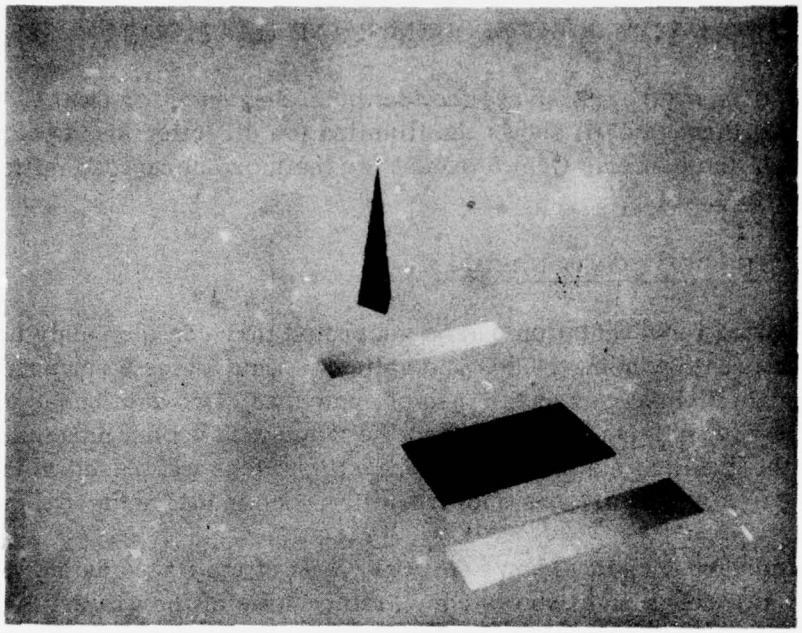
3.6 CONCLUSIONS

It was shown that contours could be simulated by applying a pre-Frame 2 computation to create tonal delineators that are not actual scene edges. This was shown to involve more computation than when edges were used and to be inapplicable to low-flight simulation.

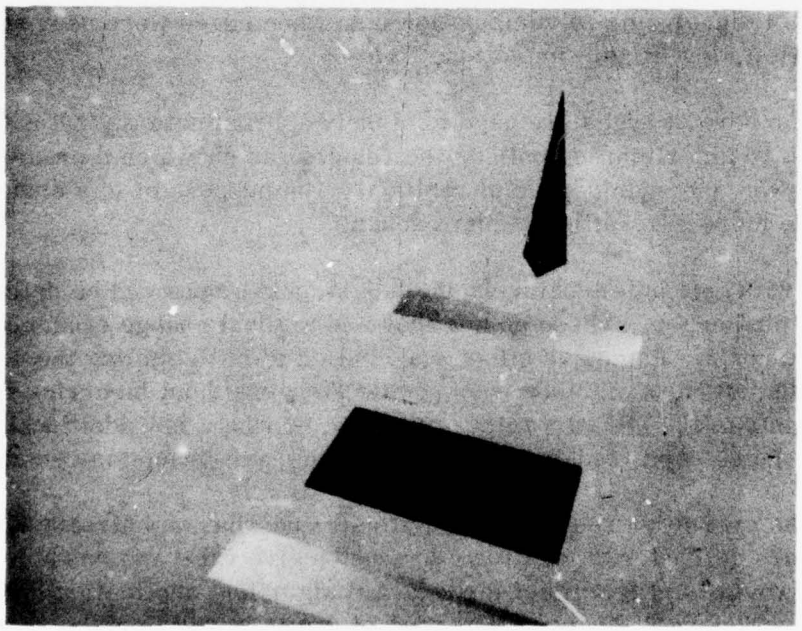
If large numbers of edges are used to closely approximate ground contours and the curvature algorithm is applied, the results, as shown on a previous contract, show a very high degree of realism. The purpose of this study was to investigate more edge-efficient approaches.

Contours with definable length, width, height, and slopes can be defined with as few as nine edges. These do not show the gradual change of slope typical of actual terrain. Extensive effort was applied to investigating the use of the gradient algorithm to simulate such gentle slopes without incurring the edge requirements of the full curvature algorithm. Using a few additional edges for transition faces corrected part of the deficiency, but problems remained.

Some of the evaluation results further illustrated what was already known—that flat contours defined with relatively few edges give valid and valuable visual cues. However, the results of the gradient algorithm application indicate that it is not applicable to the representation of spatially valid and consistent contours defined by small numbers of edges. This had not been previously known.

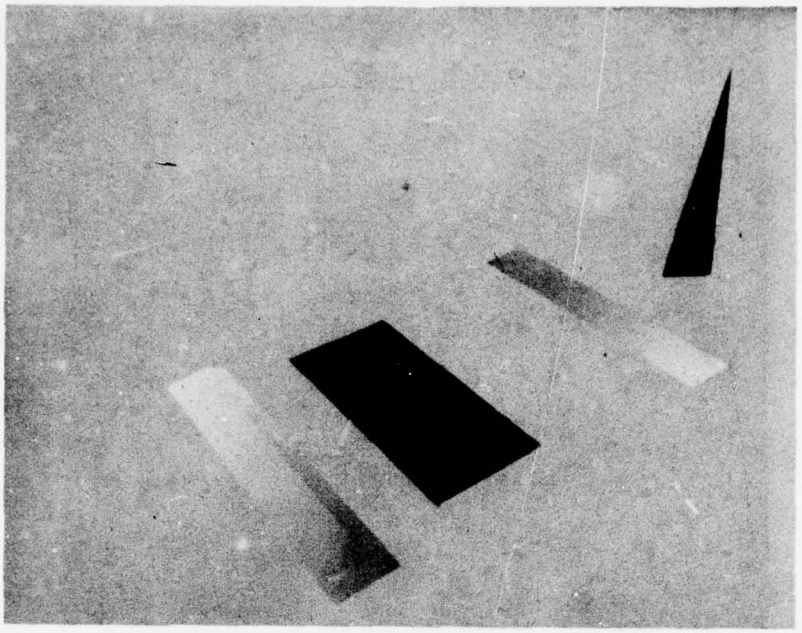


(a)



(b)

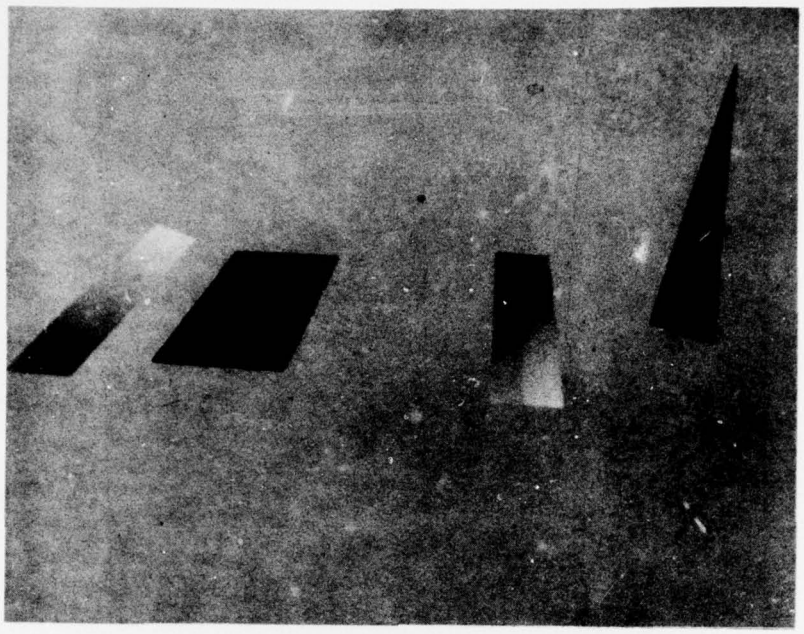
Figure 42. Cont 6 Data Base from Sequence of Viewpoints (Sheet 1 of 4)



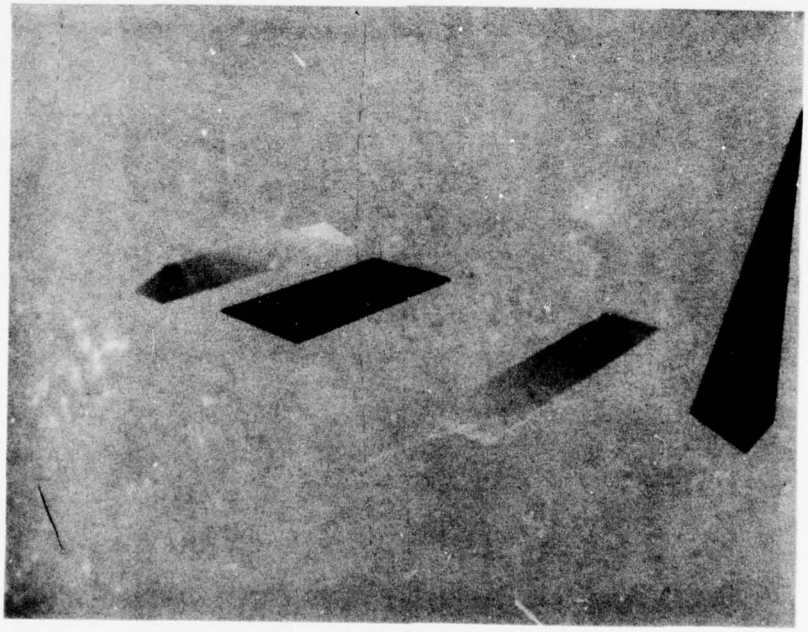
(c)

(d)

Figure 42. Cont 6 Data Base from Sequence of Viewpoints (Sheet 2 of 4)



(e)



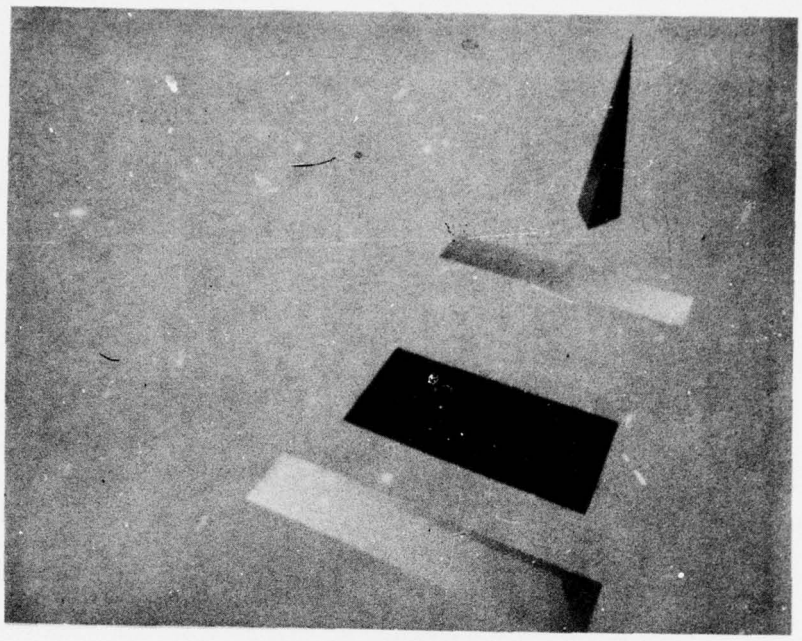
(f)

Figure 42. Cont 6 Data Base from Sequence of Viewpoints (Sheet 3 of 4)

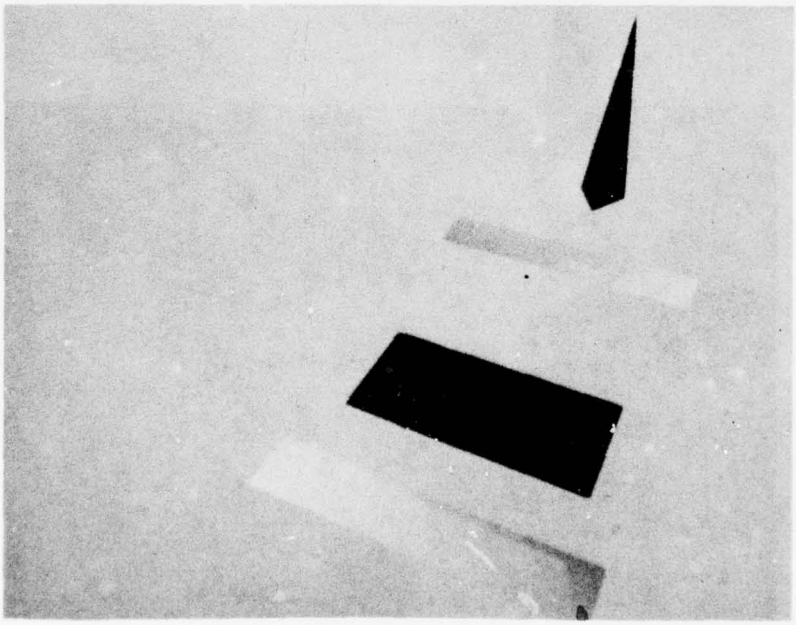
(g)

(h)

Figure 42. Cont 6 Data Base from Sequence of Viewpoints (Sheet 4 of 4)

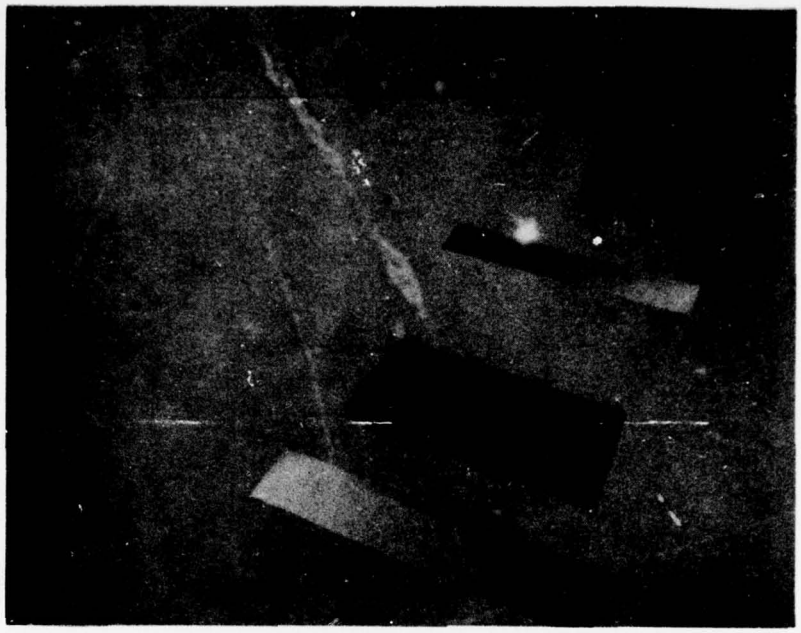


(a)

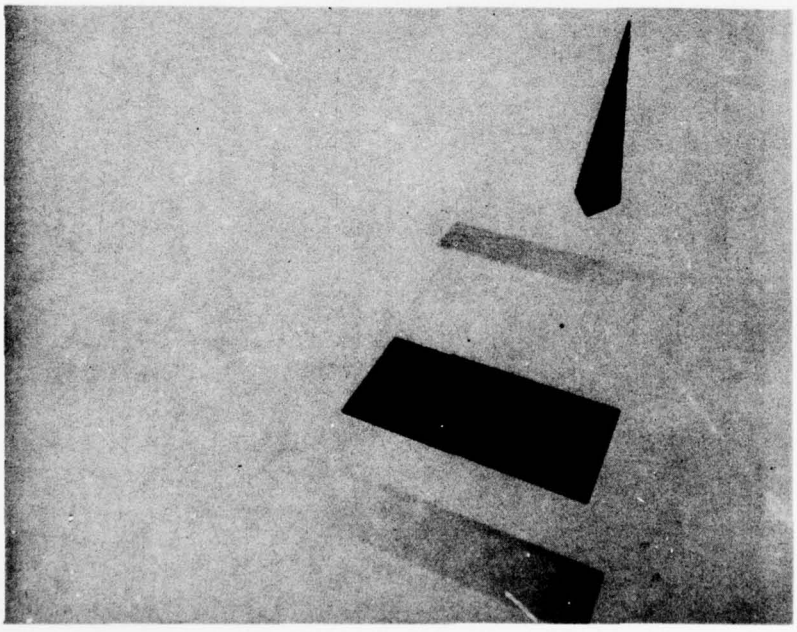


(b)

Figure 43. Cont 6 Data Base with Varying Illumination Direction (Sheet 1 of 5)

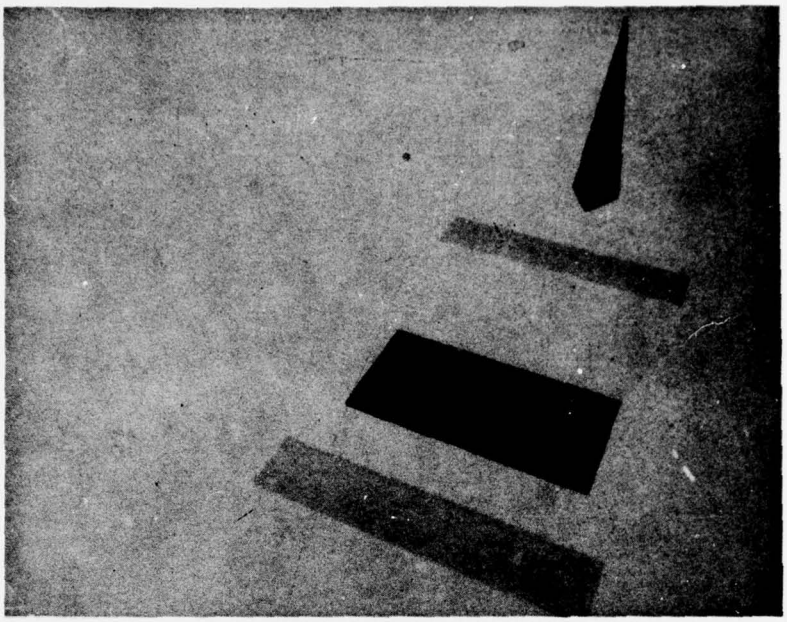


(c)

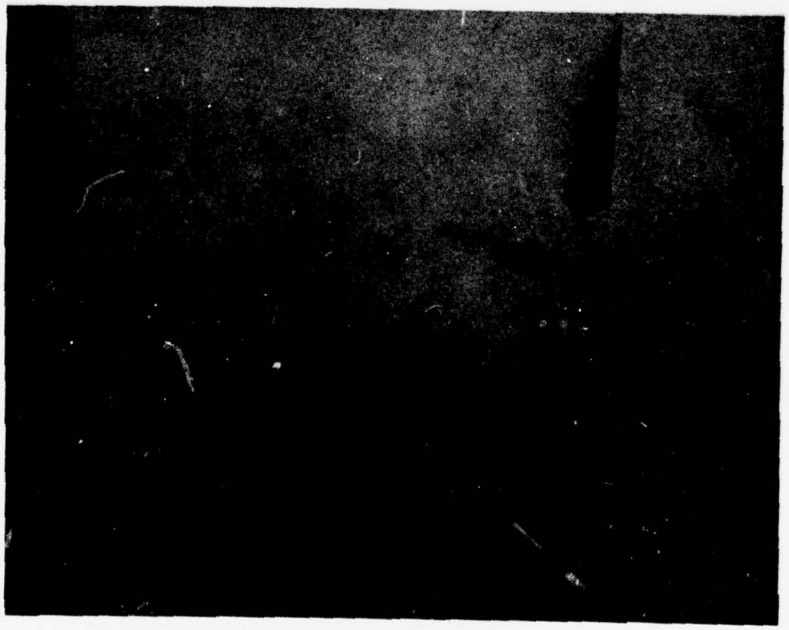


(d)

Figure 43. Cont 6 Data Base with Varying Illumination Direction (Sheet 2 of 5)

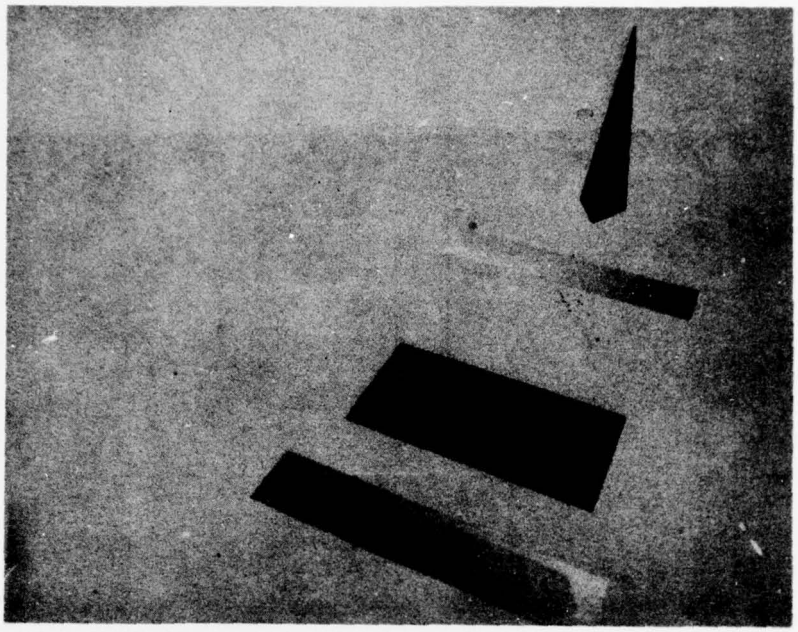


(e)



(f)

Figure 43. Cont 6 Data Base with Varying Illumination Direction (Sheet 3 of 5)



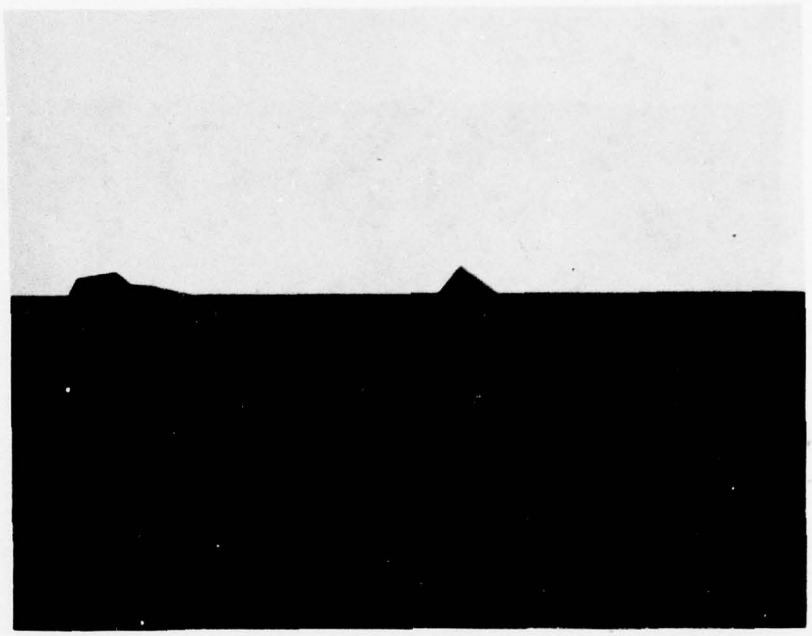
(g)

(h)

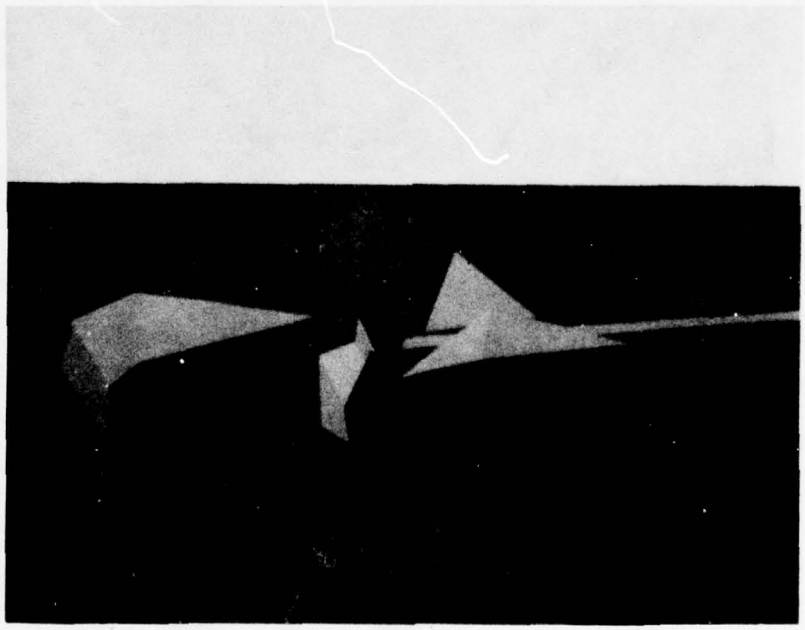
Figure 43. Cont 6 Data Base with Varying Illumination Direction (Sheet 4 of 5)

(i)

Figure 43. Cont 6 Data Base with Varying Illumination Direction (Sheet 5 of 5)



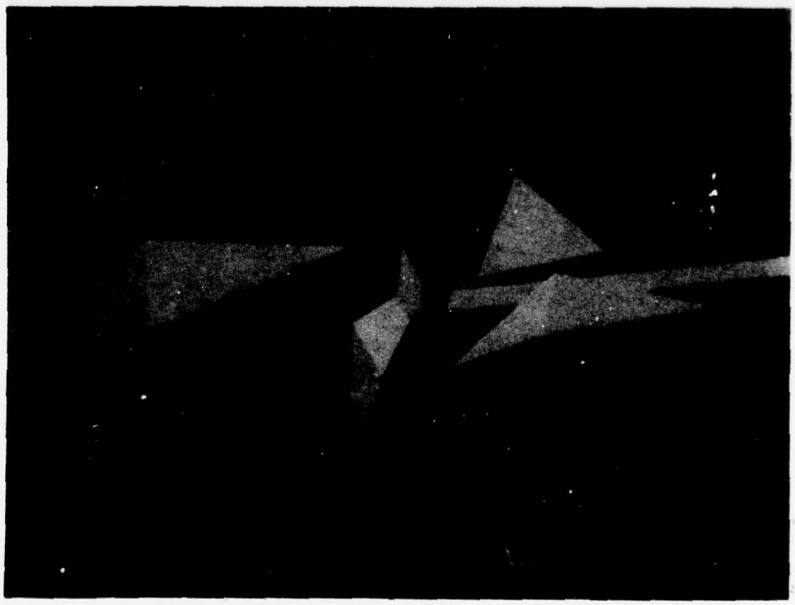
(a)



(b)

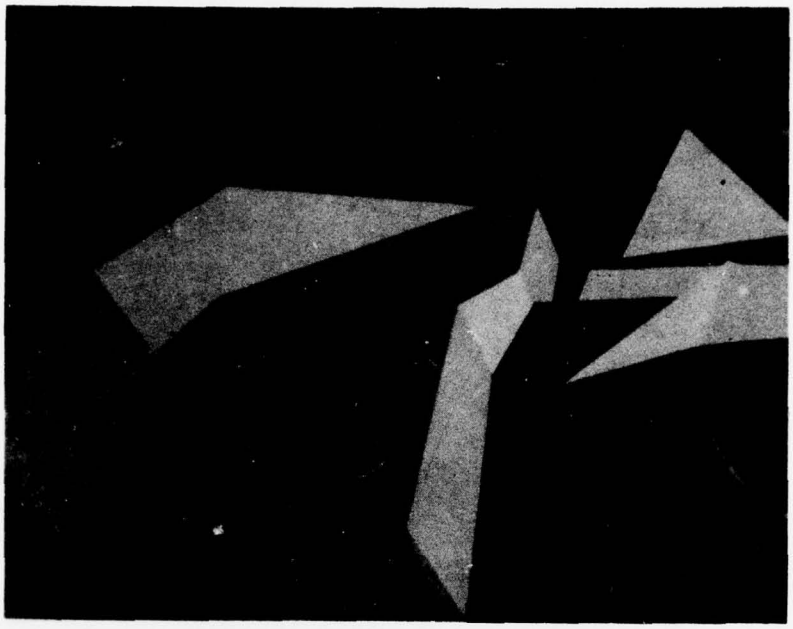
Figure 44. Connected Contours—Flat Face Version (Sheet 1 of 4)

(c)

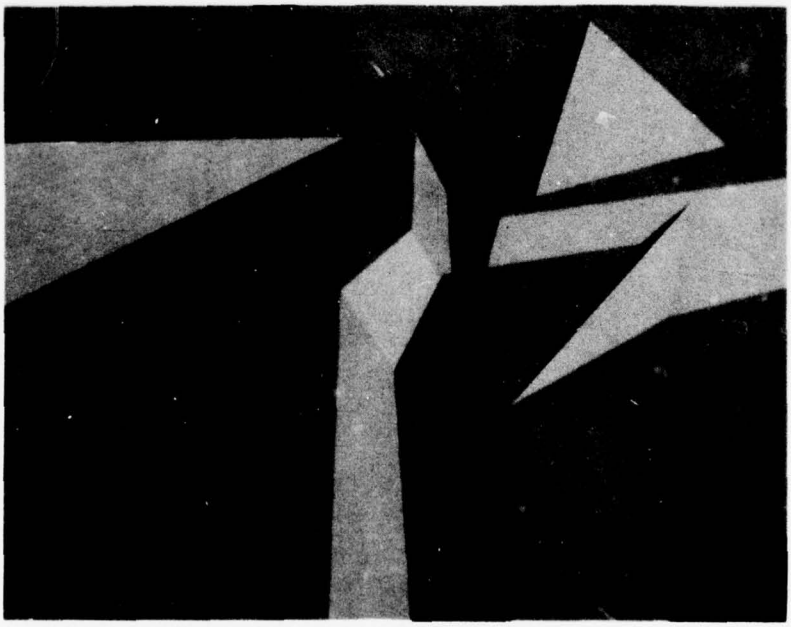


(d)

Figure 44. Connected Contours—Flat Face Version (Sheet 2 of 4)

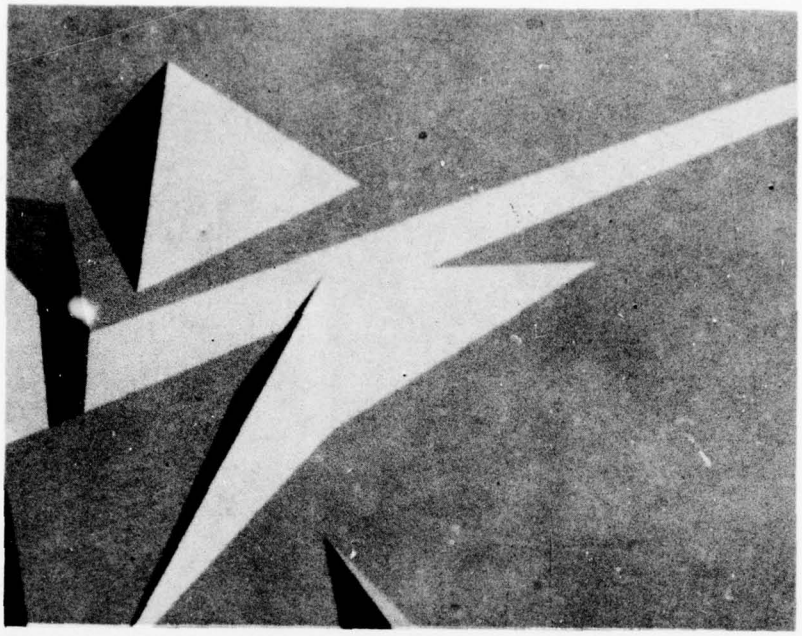


(e)

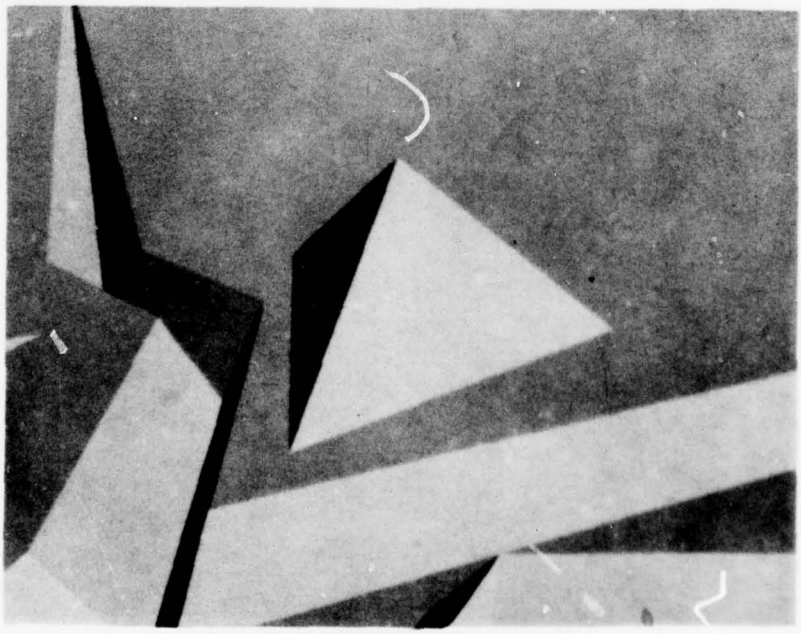


(f)

Figure 44. Connected Contours—Flat Face Version (Sheet 3 of 4)

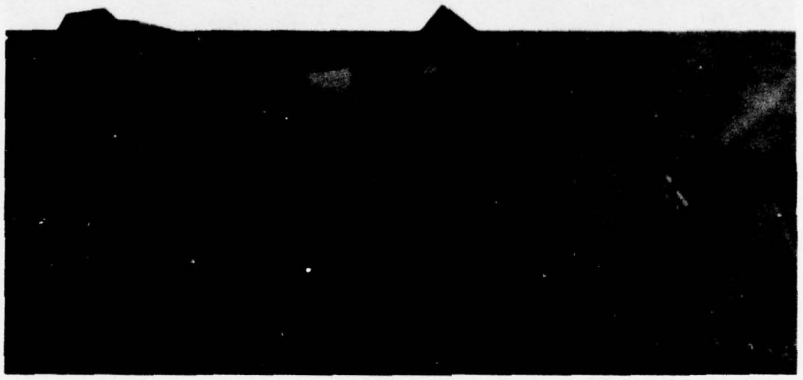


(g)



(h)

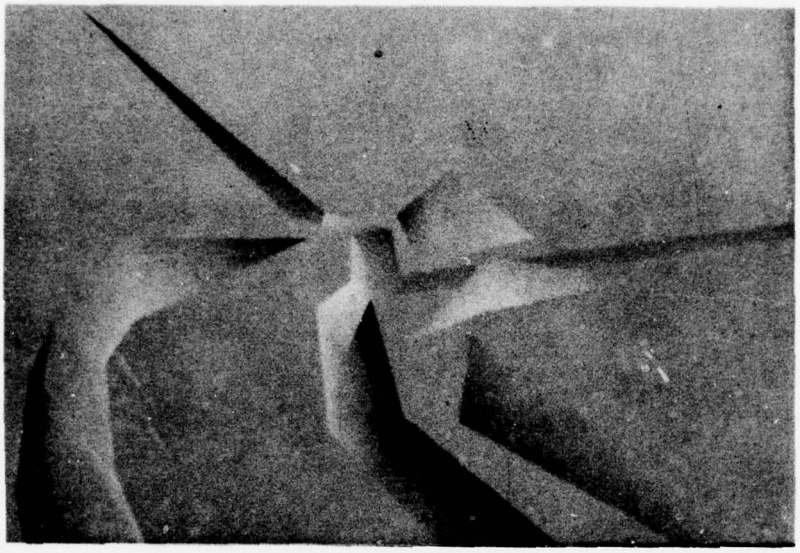
Figure 44. Connected Contours—Flat Face Version (Sheet 4 of 4)



(a)

(b)

Figure 45. Connected Contours—Gradient Version (Sheet 1 of 4)

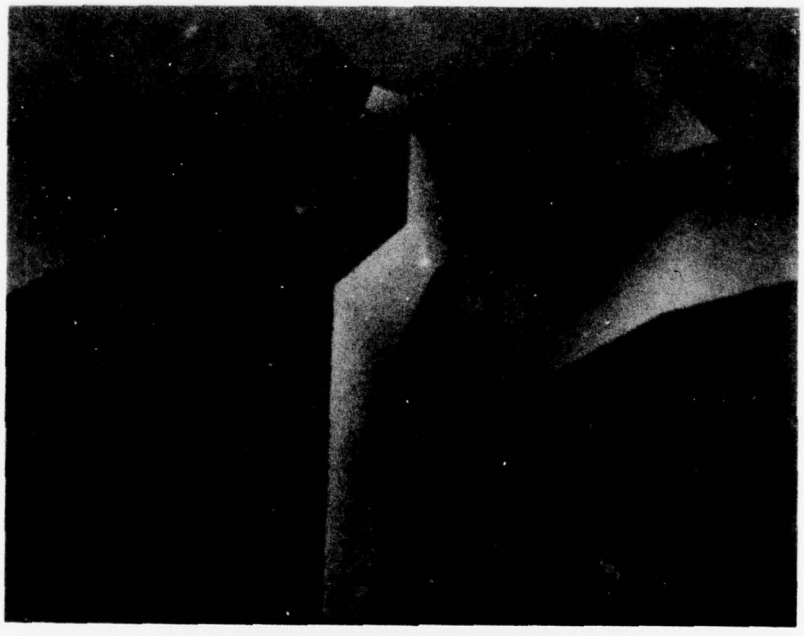


(c)

(d)

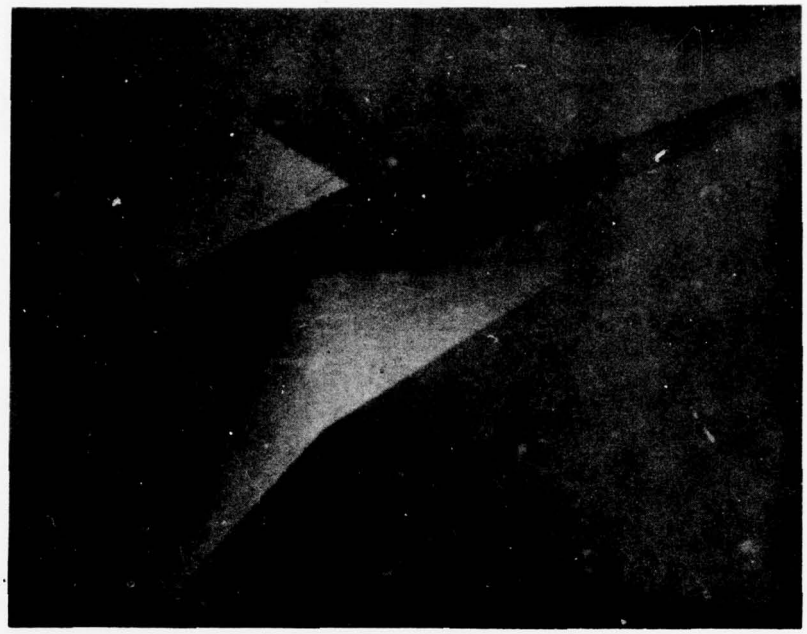
Figure 45. Connected Contours—Gradient Version (Sheet 2 of 4)

(e)

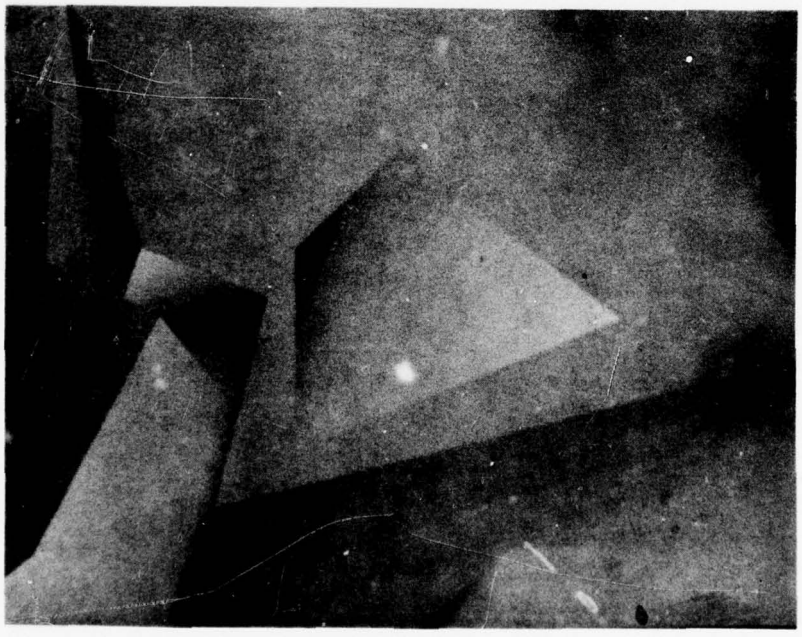


(f)

Figure 45. Connected Contours—Gradient Version (Sheet 3 of 4)



(g)



(h)

Figure 45. Connected Contours—Gradient Version (Sheet 4 of 4)

SECTION 4

SURFACE MAP TEXTURE

4.1 CONCEPT

Consider a quantitative function, q, which varies linearly across the surface of a face. Assume that when perspective transformation is applied to the face to determine its image on a display window, the pattern of numbers transforms validly. There are a number of ways in which such a pattern or function can be used.

Figure 46(a) shows a plan view of a face with the gradient vector of a function q. Assume that the plot of q versus position on the face is as shown in Figure 46(b)—a continuous, linear variation of magnitude. Such a function is used directly in implementation of the curvature algorithm. It is applied to the assigned face color to produce a smooth continuous gradation of intensity across the face. In the curvature application, the range of magnitude of the function is such as to produce only a small percentage variation in brightness across one face.

Figure 46(c) shows $q_a = q_{mod\ (1024)}$, plotted to an enlarged scale. Thinking more in hardware terms, q_a constitutes the ten least significant bits of q. In a texturing application, q_a could be used to control intensity variation of a face, over a specified range of total brightness variation. The variation would be smooth from minimum to maximum brightness with an abrupt change back to minimum.

Function q_b would be quite easy to derive from q_a — it is a q_a folded around 511 and based at zero. If this is used for quantitative variation, we no longer have the abrupt changes. Tonal modification is continuous across the entire face.

Function q_c is a logical signal produced by comparing q_b with a threshold. If used directly to control color or intensity, it would produce a row of parallel stripes (parallel on the face, not on the image).

Several functions or patterns of numbers can be combined to produce texture. Figure 46(f) shows the result of using a logical combination of q_c with a similar signal from a second function, made purposely nonorthogonal to the first function. Several such functions can also be combined in a quantitative mode. As another possibility, three functions could be used, controlling intensity of the three primary colors.

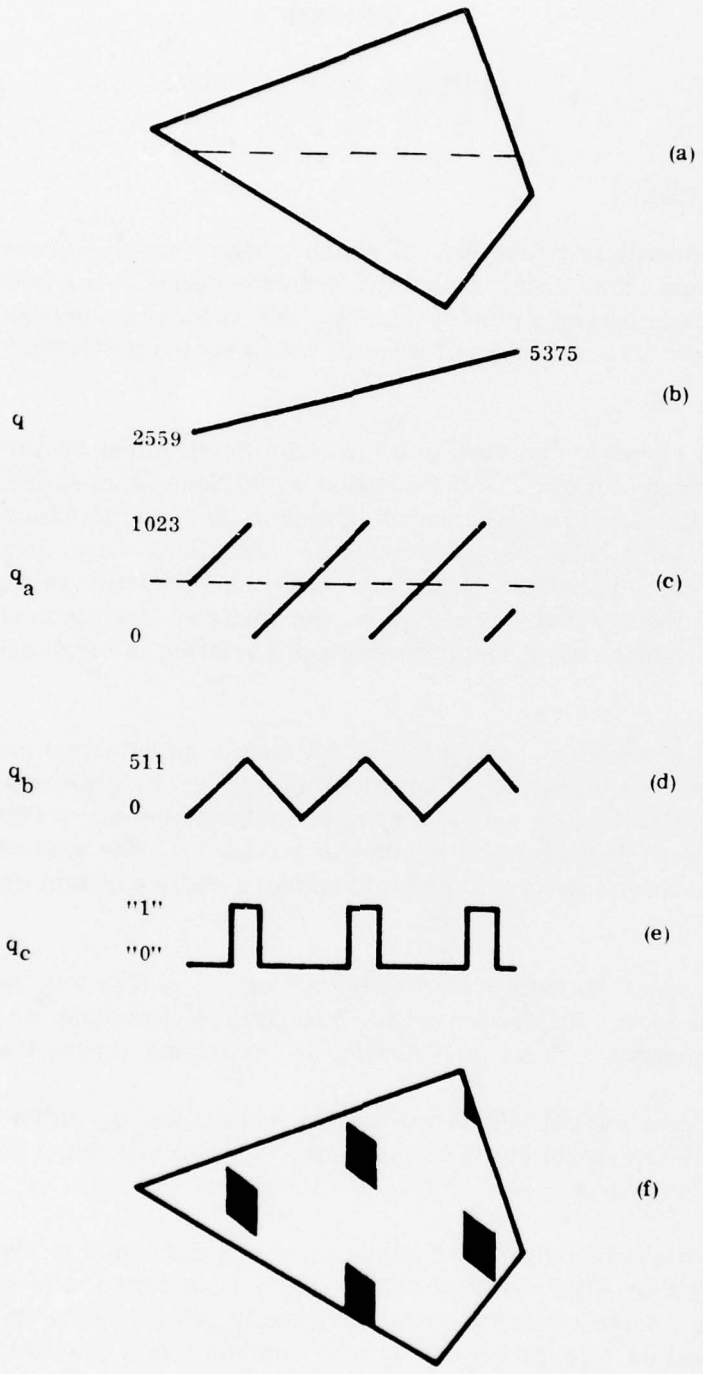


Figure 46. Linear Texture Function

Another approach involves use of sets of bits from several functions as addresses to color register memories, whose output is converted to video. This was used on the system delivered to NASA in 1964 to produce indefinitely extended surface patterns. Or, the output of a table-look-up memory can be used to modulate assigned face or surface colors.

A possible approach involves definition of several functions for each face. Their relative orientation and their orientation on one face relative to another are unconstrained. A four-bit code that is part of the face definition can specify the desired one of up to 16 different ways of using the functions—several have been mentioned above. This will provide a very powerful and versatile system for providing motion and attitude cues along large faces that are currently a uniform color.

In considering implementation details, we can consider separately the processing that produces the streams of values of "q", changing at element rate, and the subsequent steps involved with the use of these q values.

The requirements for the processing to produce the texture function values are deterministic and quite well defined. There is need for investigation of different techniques to determine which will result in minimum hardware cost.

In considering the ways in which the numbers can be used to provide texture, there is an unlimited number of possibilities. Several of these have been evaluated and are shown later in this report.

4.1.1 LEVEL OF DETAIL

A type of level of detail will have to be applied to the texturing. As a face becomes more distant, the face itself, as well as any texture features on it, become very small on the screen. If the magnitude of the modulation of the face color is reduced with increasing distance, the visibility of texture detail will gradually decrease, very much as it actually does in the real world.

The following numerical example will provide additional insight into the modulation map use of the linear functions as well as describe how level of detail is implemented with it.

4.1.2 MODULATION MAP

Figure 47 shows values of Y at 32-foot intervals. These are not handled as edges. They can be thought of as dividing the surface into parallel strips or zones. To the left are the values of Y; 7296, 7328, etc. To the right are the values of Y in binary notation. Assume we have a two-bit, table-look-up memory as shown in Table 3.

	12	11	10	9	8	7	6	5	4	3	2	1	0	
7552	1	1	1	0	1	1	0	0	0	0	0	0	0	
7520 (4)	1	1	1	0	1	0	1	1	0	0	0	0	0	
7488 (2)	1	1	1	0	1	0	1	0	0	0	0	0	0	(3)
7456 (9)	1	1	1	0	1	0	0	1	0	0	0	0	0	
7424 (3)	1	1	1	0	1	0	0	0	0	0	0	0	0	(5)
7392 (4)	1	1	1	0	0	1	1	1	0	0	0	0	0	
7360 (2)	1	1	1	0	0	1	1	0	0	0	0	0	0	(3)
7328 (9)	1	1	1	0	0	1	0	1	0	0	0	0	0	
7296 (3)	1	1	1	0	0	1	0	0	0	0	0	0	0	(5)

Figure 47. "Y" Map

Table 3

Level 1 Memory Contents

In	Out
00	3
01	9
10	2
11	4

Use bits 5 and 6 of the binary Y as input to this memory, and let the output be a number which in some manner characterizes the zone. The assigned numbers are shown inside the circles on the illustration.

If the numbers obtained from the map output are used to modulate the assigned color of the surface background or of surface faces which it is desired to texture, we will have a texture pattern of stripes covering the entire gaming area. The stripes are 32 feet wide; this follows from the fact that the 5th bit of Y was used as the least significant bit of the address. The pattern repeats every four stripes. This follows from the fact that we use a two-bit segment of Y to address the memory. A repetition period of 1024 zones, achieved by using a 10-bit address, would be more suitable for an actual system.

As we consider portions of the surface more and more distant from the viewer, the 32-foot interval becomes fewer and fewer element dimensions on the display. This led to some of the more disturbing effects of the NASA implementation. The concept developed in pre-1976 IR&D effort handles this as follows.

Consider a one-bit table-look-up map as shown in Table 4.

Table 4

Level 2 Memory Contents

In	Out
0	5
1	3

Now if we use only bit six of Y as an address, we have 64-foot zones, characterized by the numbers shown circled to the right of Figure 47. Note that the characteristic number of each 64-foot zone is the average of the numbers of the two 32-foot zones forming it. For this example, the original 32-foot zones can be considered as the Number 1 level of detail, and the 64-foot zones as level of detail 2. We switch to Number 2 when the size of the zone on the display becomes less than some assigned lower limit. Thus we never get zones smaller than an element, and the contrast will gradually decrease with distance in a natural manner. The "averaging" described above is not to be used blindly. In some cases, such as when the map contents are determined by some mathematical function, this will give very poor results. The function itself must be applied, with reduced contrast, to the lower detail maps to achieve the desired results.

Note that the entries in the map memory may be assigned by a random number generator, or they may be chosen by the modeler to achieve any effect desired.

A pattern of stripes may be fine for some special applications but is certainly not adequate in general. As a first step in extending it, consider a map in the X-direction defined just as the Y map, with memory contents set to: 1, 7, 6, 3. For any location on the surface add the contents of the X and Y maps. Figure 48 shows the result—an indefinitely repeating array of 16 cells.

This will still provide directional orientation cues which may be undesirable. We can extend the map concept as follows.

Define two more orthogonal pairs of maps in like manner, rotated 60 degrees and 120 degrees relative to the X-Y axes. Figure 49 shows these six maps superimposed. A circular section is shown to emphasize the way in which the orientation of the axes is obscured. Not only is there no apparent repetition pattern; it can be shown that due to the irrationality of sin (60 degrees), no repetition pattern exists. If we add numbers from the six maps, each bounded region of the surface has its own characteristic number.

In the discussion of the use of these numbers, keep in mind that in an actual system a 10-bit address will typically be used for the table-look-up memory, so that even along one axis there will be a set of 1024 numbers of strips before there is repetition.

Each face, in addition to being assigned a color, will carry bits defining the applicability of the modulation numbers. A face representing part of a river may be blue with texture inhibited. A farm field may be brown, with numbers from the "Y" map used full strength to modulate the assigned color, and possible "X" numbers used at 25 percent strength, to give an impression of plowed field with some texture along the rows. Similar techniques can simulate waves in the ocean. Where general texture is desired with no directional cues, the full map output can be used to provide interior features in large surface faces.

In addition to the use of modulation numbers to modify the brightness of an assigned color, they may be applied to only one or two of the primary colors, or to the colors in different strength, so the texture also includes color variation.

The Surface Modulation Map provides an extremely versatile tool. It can texture an indefinite area of terrain. The nature of the texturing can be varied from one face to another. Features are incorporated to eliminate the directional cues, the repetition pattern, and the scintillation in the distance which were characteristic of the earlier terrain map approach.

	1	7	6	3	1	7	6	3
4	5	11	10	7	5	11	10	7
2	3	9	8	5	3	9	8	5
9	10	16	15	12	10	16	15	12
3	4	10	9	6	4	10	9	6
4	5	11	10	7	5	11	10	7
2	3	9	8	5	8	9	8	5
9	10	16	15	12	10	16	15	12
3	4	10	9	6	4	10	9	6

Figure 48. "X" and "Y" Maps Superimposed

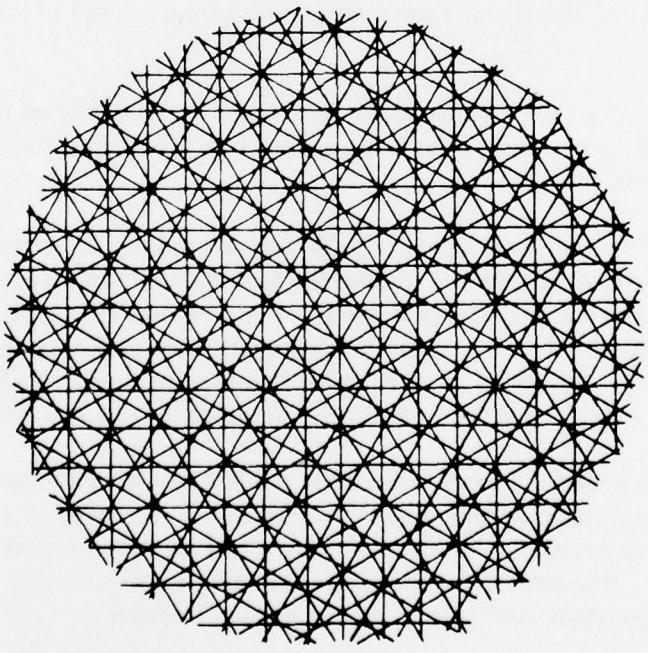


Figure 49. Pseudorandom Pattern for Perspective Texture

4.2 EVALUATION

A software simulation model was programmed to implement surface map texturing in a nonreal-time system. A high degree of flexibility was incorporated in varying program parameters to evaluate a variety of possible map patterns. A set of six maps with 30-degree spacing, such as was discussed earlier, was defined. The map entries were filled using a random number generator. Figure 50 shows a plan view of the map texture test scene used for evaluation. The initial parameters were as follows. Zone width was set at 8 feet. Minimum elements per zone to initiate a lower level of detail was set at 8 elements. Repetition period was set to 64 zones (limited by computer core).

The background, where no faces are defined, was modulated by all six maps. Face 14 has exactly the same texture modulation as the background, but it is applied to a different assigned color. Face 13 uses a subset of only two of the six maps. Face 12 uses a different subset of two. Face 11 uses all six maps, but is applied to only one of the three primary colors, so texture will consist not just of a variation in brightness, but of a color variation.

Face 9 uses only the X map, with quantitative rather than table-look-up texture. Texture spacing was set to 60 feet. Face 10 uses the X and Y maps in combination, in quantitative mode, with 25-foot spacing.

Face 15 is the same as Face 14, except with a level of detail bias of "one." For any distance, it will be shown at the next lower level of detail than the background.

Figure 51 shows a view of the scene with parameters defined as above. Modulation strength is set rather high to clearly bring out the effect of the different modes of operation.

Figure 52 shows the scene from a different viewpoint, to include the surface from close up to the horizon. Texturing is disabled, and we see the familiar "features against an artist's backdrop" effect, which illustrates the need for texturing. Figure 53 is the same but with texturing. There are two differences from Figure 51. Program modification freed some core, so the repetition period is 128 zones, and the modulation magnitude was reduced.

Figure 53 was produced primarily to evaluate the effect of level of detail handling. The bottom of the scene is at level 1. The region at the near side of the far tower is level 2. The lowest level, reached somewhere before the horizon, is 7. The effectiveness of the technique is indicated by the fact that one cannot determine just where the transitions occur.

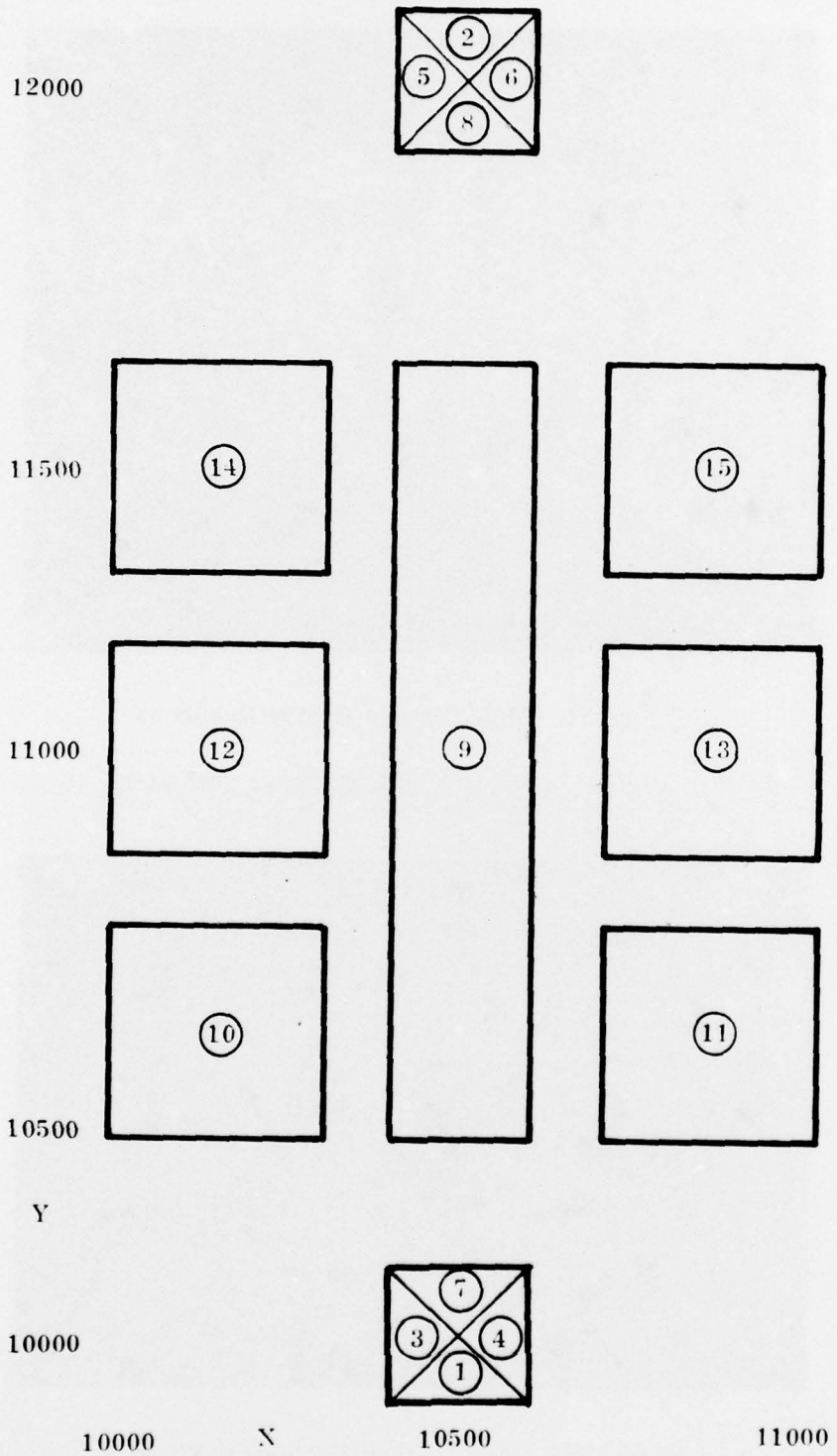


Figure 50. Map Texture Test Data Base

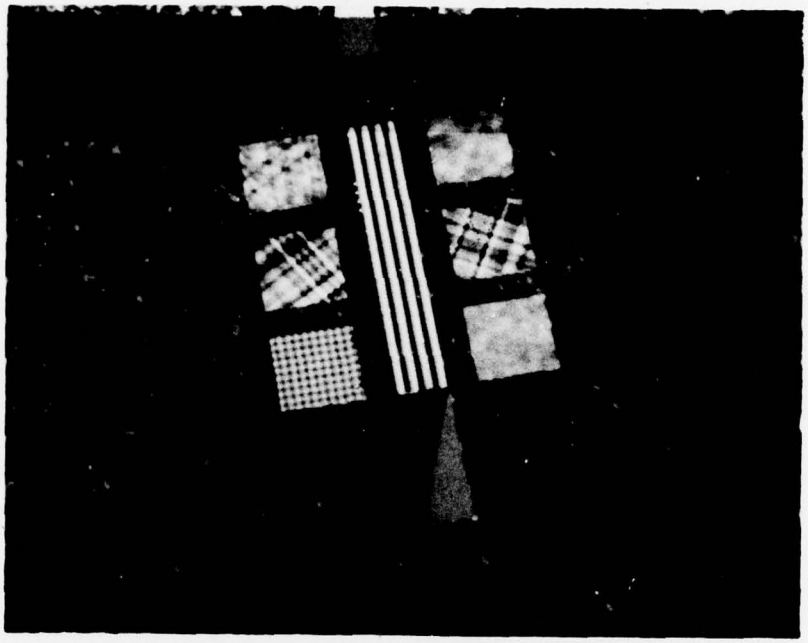


Figure 51. Map Texture Evaluation Scene

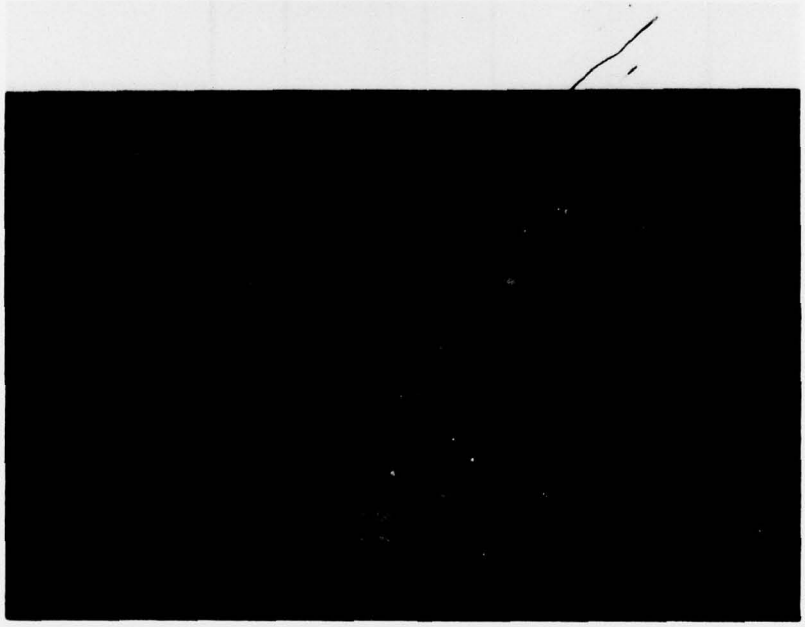


Figure 52. Map Texture Evaluation Scene

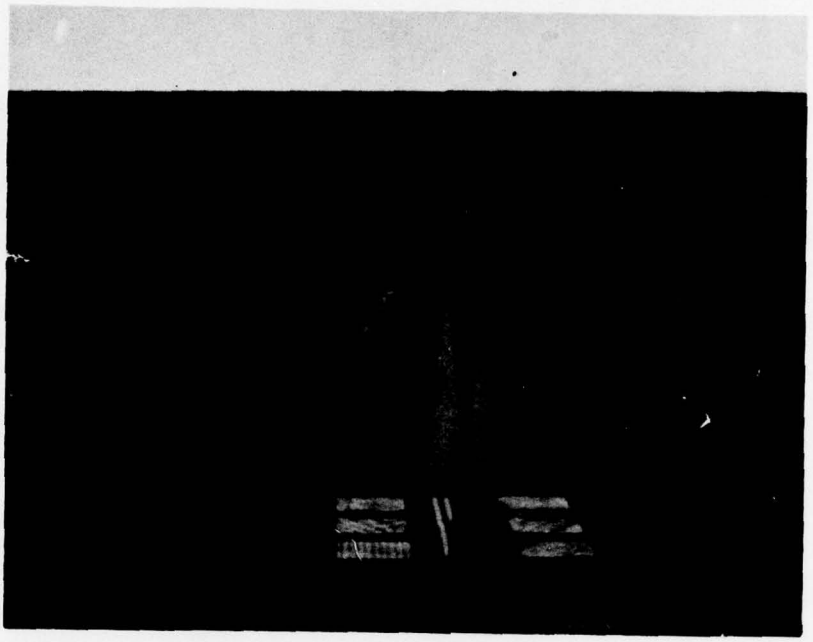


Figure 53. Map Texture Evaluation Scene

To further examine the effect of parameter variations, the zone width was changed from 8 feet to 16 feet, and the spacing on face 10 was changed to 50 feet. Figure 54 was then produced. Finally, with no further parameter changes, the viewpoint was changed, and Figures 55 (untextured) and 56 (textured) were made. This illustrates the spatial consistency of the texturing.

Figure 57 was made as an initial attempt to simulate rolling waves. It uses the same data base as the previous figures with a change in the application of texture. The contents of the X-map memory, rather than being filled by a random number generator as before, was filled by a sinusoidal function. The magnitude of the X-map modulation was increased, and the magnitude of the other five maps was decreased.

4.3 MAP TEXTURE—CONCLUSIONS

The previous scenes effectively validate the algorithms and indicate that the texture produced will be very effective in providing visual cues now missing from CIG-generated visual scene simulation. Current effort to extend this technique to nonsurface faces will provide very sensitive indications of surface contours and make CIG visuals even more valuable in a variety of training requirements.

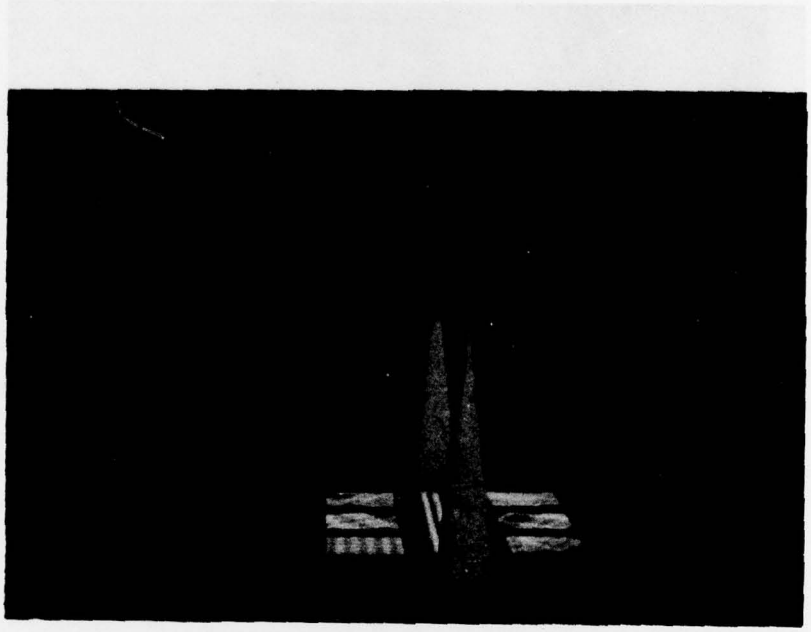


Figure 54. Map Texture Evaluation Scene

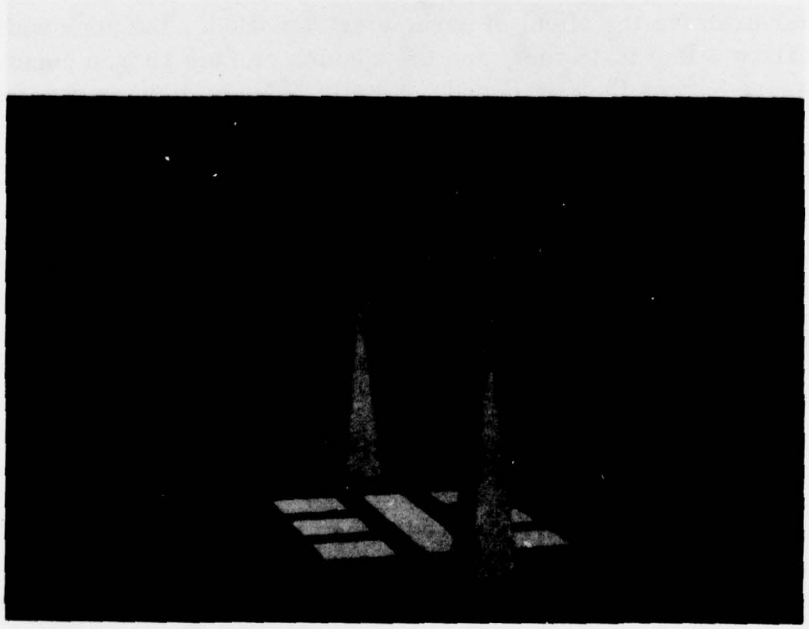


Figure 55. Map Texture Evaluation Scene

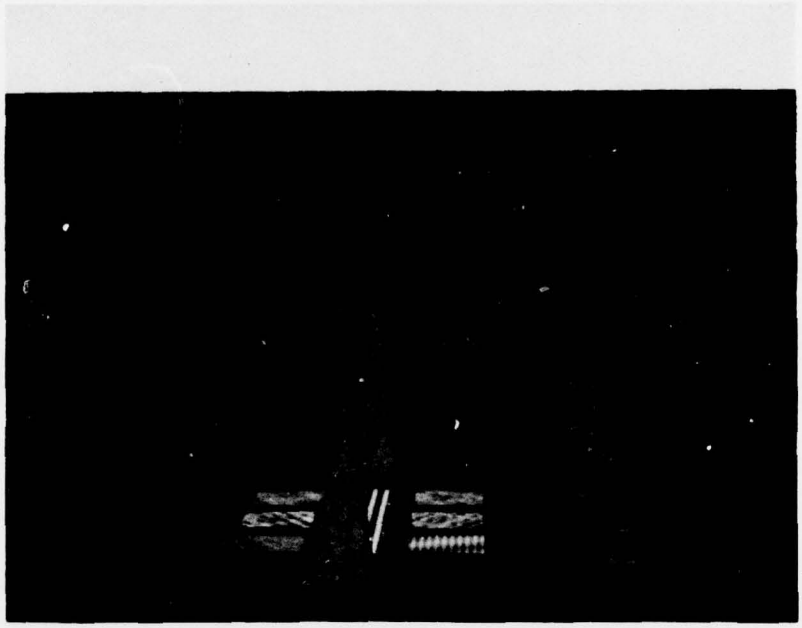


Figure 56. Map Texture Evaluation Scene

Figure 57. Sinusoidal Texture Function

SECTION 5

FEATURE GENERATOR

5.1 BACKGROUND

This study has as its purpose the improvement of visual cues in applying CIG to training missions. The three specific areas discussed are elliptical and spherical features, contours and surface-map texture. The feature generator concept is applicable to all these and to standard systems with standard edges as well.

Assume the existence of a CIG system with its fixed capacity for edge generation, point-light generation, and generation of other types of features (e.g., spheres). Assume that the amount of detail that can be generated and displayed on any single scene is sufficient that visual cue requirements for every portion of the training mission could be adequately met. It is probable that some existing systems come close to meeting this assumption. Nevertheless, on an extended mission there are large portions of the terrain very sparse in detail and cues.

In a CIG system, the major portion of the cost is devoted to providing the per-scene capacity—the potentially visible edges, etc. If a trainee is flying between airports and is in need of more scene detail while this portion of the system is generating only a few dozen edges, it is being largely wasted when it could be contributing to the training task. A goal of the earlier processing, the less complex functions, which can be done at a far slower rate, might be stated as keeping the later portion busy, producing near its capacity for all scenes where any benefit can be dervied from the additional detail.

Approaches to this goal are represented in the data management systems of the ASPT and AWAVS CIG systems. Such systems have a large, slow, mass memory with capabiliy typically in the hundreds of thousands of edges. They have an active environment of a size sufficient to make effective use of the potentially visible edge capacity. This is on the order of 10,000 edges for a system with 2000 potentially visible edges, increasing proportionately for higher detail systems. The data management system continually updates the contents of the active environment memory as the mission progresses. It might seem this approach would have the capability of meeting the goal stated above. Consider, however, a system which has a 1500-mile-square gaming area. Assume it is designed for low-level missions and it is desired to have a texture or contour feature, on the average, for each 64-foot square. This is a total of 1.5 by 10^{10} such features, and each may require several edges or the equivalent in data to define.

5.2 POSSIBLE SOLUTION

As an example of the feature-generator concept, start with a system containing a standard data management system capable of processing the deterministic objects and faces in mass memory and load them into the active environment memory as the flight proceeds. Looking toward the future, it would also store and process circular and spherical features representing specific cultural or terrain objects (e.g., water towers and settling tanks). Now assume as an addition to the front-end data management system a set of functions with pseudo-random characteristics to control the function generation process. We might have tree-generation capability, with these functions controlling the tree definition. Their locations, diameters, heights, colors, shapes, etc., would vary with controlled statistics. This is a similar concept to the pseudorandom noise used for texturing-type capability in radar display simulation, but handling it as described above retains perspective validity as required for CIG.

An absolute requirement for such a system can be expressed as follows. If we see a medium-sized, blue spruce at the southeast corner of the cemetery when flying north at the start of the mission, and we return an hour later, the feature-generation system must recreate this tree with the identical definition as before.

With this approach, there is no limit to the number of features that can be seen during the course of a mission. The feature generator creates them when they are potentially visible; they do not burden memory or any other portion of the system when they are not needed.

The concept can be applied to ground shapes defined with edges or to contours defined with edges. It can be applied to small-scale texture created with the system's point-light generator. It can be applied to ellipses and spheres. Clouds, such as shown in Figure 14, could be scattered over the entire sky.

The statistics of ground texture could be varied based on surface delineators defining various regions. Such statistical information could be prepared by DMA or perhaps derived from existing DMA data descriptors. This type of information might be defined in a transform domain—frequency or sequency—and be translated into deterministic spatial terms by the feature generator.

Considerations of how to use most effectively some of the results of the effort under this contract led to the feature-generator concept as described previously. It appears to be feasible, powerful, and valuable. There has been no opportunity to develop detailed approaches for additional evaluation.

SECTION 6

CONCLUSIONS AND RECOMMENDATIONS

6.1 ELLIPTICAL AND SPHERICAL FEATURES

These were shown to be very valuable for adding to scene realism, as well as for the originally intended use as surface detail. Adding them to a system with current design and architecture would involve a moderate number of boards, as covered in the hardware estimate. However, the net cost to include these features as part of the capability in a system redesign is projected to be much lower. It is recommended that these features be part of the capability of future CIG systems. Ellipsoids should be included.

6.2 CIRCULAR FEATURE EQUIVALENT EDGES

The quantitative measure of the ability of a CIG system to provide scene detail has generally been in terms of its edge processing capacity. One circular feature is equivalent to some number of edges in its ability to provide visual cues and simulated detail. Some of the evaluation scenes in this report provide numbers for the equivalence ratio for specific cases.

Figure 3 shows a bomb circle produced in the ASPT system. To produce a set of four circles as shown, approximating each with 16 edges, requires processing and display of 64 edges. Figure 4 shows a similar scene, formed of four circular features. For this scene, we can state that one circular feature is equivalent to 16 edges.

The largest circle on Figure 4, spanning about 233 scan lines, is actually formed on the display of 466 segments (per scan line edges). To get an equally perfect image of a circle would thus require using 466 edges.

Figure 5 shows some cylindrical storage tanks produced with 48 edges required for each tank. Figure 6 shows similar tanks formed of two circular features and four edges each. Each feature thus replaces 22 edges.

Figure 10 also contains some of the two-feature, four-edge tanks, along with shadows formed of a single circular feature each. If formed with edges, sixteen or so would be required for each shadow.

In Figure 10, the top of the water is formed of a single spherical feature. The hemispherical top of the silo, for comparison, is formed using 68 edges.

Consideration of the above examples leads to the conclusion that, where it is appropriate to use circular/spherical features in a scene, each is equivalent to 16 or more edges.

6.3 CONTOURS

Extensive evaluation into the approach of using the gradient algorithm to improve realism of contours produced with small numbers of edges indicated this technique to be of minor merit. The surface-definition approach to contour definition involves lower capability and greater computational requirements than the use of edges. It is thus completely without merit.

Minimum-edge contours produced without the gradient algorithm provide effective and valid visual cues of the type desired from contours, although the obvious lack of gradual transition of slope decreases realism.

Fully realistic contour depiction is possible by using sufficient edges to closely approximate terrain contours within the requirements of the curvature algorithm No other approach is currently known to do this.

In summary, fully valid contour cues can be produced in two ways, realistically with many edges, or with very few edges and lacking in realism.

6.4 SURFACE MAP TEXTURE

Although this report does not include a cost estimate for implementation of surface-map texture, it can be stated that it will be a rather costly addition. On the other hand, it gives promise of providing extremely valuable and valid texture cues over an unlimited gaming area. It will be even more effective when current effort extending it to nonsurface faces is completed. It should be specified for systems where the value of this texture justifies the added system cost.

6.5 FEATURE GENERATOR

This concept, applicable to edges as well as features produced by other means, has great potential value. Its application is not limited to systems of the future. Its nature is such that retrofitting this capability into existing systems is quite feasible.

It is recommended that development of the concept be continued, that a study be initiated to develop and evaluate algorithms to implement the concept, and that estimates for hardware implementation be prepared.

☆U.S. GOVERNMENT PRINTING OFFICE: 1978-771-122/13